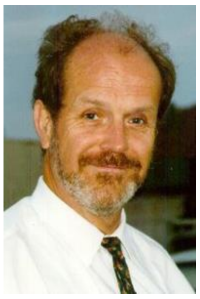
UDK 004.75

DATA PROCESSING IN HIGH-PERFORMANCE COMPUTING SYSTEMS

A. Adamov
Director, Center for Data Analytics Research (CeDAR)

A.I. Buranbaeva
Associate Professor

S.I.Gindin
Associate Professor of Information and Computer Systems Department


M. Glesner
Professor, IEEE Fellow, head of Microelectronic System Research Group

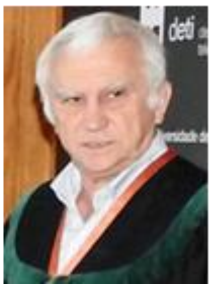
I. Horton
Author of books dedicated to programming languages C/C++/Java and software development

A. Keevallik
Professor of Tallinn University of Technology

A.D. Khomonenko
Professor, the head of Information and Computer Systems Department

Y.G. Klyuyeva
Senior lecturer of Information Technology and Security Department

I. Skliarova
Auxiliary Professor of Aveiro University


V. Sklyarov
Professor, honorary doctor

V.V. Yavorskiy
Professor of Information Technology and Security Department

Center for Data Analytics Research (CeDAR), Baku, Azerbaijan
Kazakh Academy of Transport and Communications (Institute of Information and Computer Technologies, Ministry of Education and Science of Kazakhstan), Almaty, Kazakhstan
Information and Computer Systems Department of Emperor Alexander I St. Petersburg State Transport University, St. Petersburg, Russia
Darmstadt Technical University, Darmstadt, Germany
Stratford-upon-Avon, England
Tallinn University of Technology, Tallinn, Estonia
Karaganda State Technical Universit, Karaganda, Kazakhstan

Aveiro University, Aveiro, Portugal
E-mail: skl@ua.pt

A. Adamov

Director, Center for Data Analytics Research (CeDAR), School of IT & Engineering at ADA University, General chair of the 14th IEEE International conference on Application of Information and Communication Technologies (details about upcoming conference are available at: www.aict.info), Baku, Azerbaijan. The main research area is data science and analytics.

A.I. Buranbaeva

Associate Professor of Kazakh Academy of Transport and Communications (Institute of Information and Computer Technologies, Ministry of Education and Science of Kazakhstan), Almaty, Kazakhstan. The main research area is application of big data.

S.I. Gindin

Associate Professor of Information and Computer Systems Department of Emperor Alexander I St. Petersburg State Transport University, St. Petersburg, Russia. The main research area is big data technologies, databases, and modeling of IT systems.

M. Glesner

Professor, IEEE Fellow, head of Microelectronic System Research Group at Darmstadt Technical University, honorary doctor of Tallinn Technical University (Estonia), Bucharest Polytechnic University (Romania), Mongolian Technical University (Mongolia) and the University of Liepaja (Latvia), Darmstadt, Germany. The main research area is in the scope of embedded systems design, high-level synthesis and physical design.

I. Horton

He worked for IBM for more than 30 years and was responsible for implementing large systems in automotive, aerospace, and ship building. From 1972 to 1976 he was project manager for the design and installation of the IBM systems at Kamaz (Naberezhnye Chelny, Russia). After retirement from IBM he started writing books about programming.

A. Keevallik

Professor of Tallinn University of Technology, honorary doctor of Helsinki University of Technology (Finland) and Caucasus University (Georgia), Tallinn, Estonia. He is author and editor of 10 books. The main research area is digital systems decomposition based on divide-and-conquer algorithms.

A.D. Khomonenko

Professor, the head of Information and Computer Systems Department of Emperor Alexander I St. Petersburg State Transport University, St. Petersburg, Russia. He is author of 25 books. The main research area is big data technologies, databases, and modeling of IT systems.

Y.G. Klyuyeva

Senior lecturer of Information Technology and Security Department of Karaganda State Technical University. Karaganda, Kazakhstan. She is author of 3 books. The main research area is big data technologies.

I. Skliarova

Auxiliary Professor of Aveiro University, Aveiro, Portugal. She is author of 6 books written in English and Chinese. The main research area is data processing, FPGA-based designs, hardware accelerators and combinatorial optimization.

V. Sklyarov

Professor, honorary doctor of Tallinn University of Technology (Estonia), Aveiro, Portugal. He is author of 23 books written in Russian, English and Chinese. The main research area is reconfigurable computing, data processing, and hardware accelerators.

V.V. Yavorskiy

Professor of Information Technology and Security Department of Karaganda State Technical University, Academician of RANS.. He is author of 14 books. The main research area is big data technologies.

Abstract. The paper integrates the results of a large group of authors working in different areas that are important in the scope of big data, including but not limited to: overview of the basic solutions for the development of data centers; storage and processing; decomposition of a problem into sub-problems of lower complexity (such as applying divide and conquer algorithms); models and methods allowing broad parallelism to be realized; alternative techniques for potential acceleration; programming languages; and practical applications.

Keywords: big data, data processing, data centers, MapReduce concept, hardware accelerators.

Introduction. Engineering and scientific applications have become more data intensive [1] demanding emerging high-performance interactive frameworks. Existing processing techniques may be applied at different stages from upper levels aimed at developing data processing centers with the primary focus on software for high performance accelerators that involve additional hardware components. Hence, a range of diverse types of expertise are required, which resulted in the large group of authors being involved. This allowed the scope of big data and acceleration at lower levels to be combined and reviewed. The importance and complexity of big data [2] forces computer engineers to seek new solutions for processing them. Processing speed still continues to rise exponentially while memory bandwidth increases at a much slower rate [3]. Working with big data requires huge storage, extensive processing capability, and high-performance communications for data exchange [3]. It is explicitly stated in [2] that based on previous knowledge we need to introduce new techniques for data manipulation, analysis, and mining. The experience of the authors of the paper falls into three different areas: big data management; high-performance computing systems and hardware accelerators; practical applications necessitating working with big data. It is difficult or perhaps impossible to discuss all the potential features in the areas selected above. Thus, particular topics (given sections of the paper) have been chosen in accordance with the authors' expertise and they are dedicated to: basic solutions for the development of data centers; a review of potential hardware accelerators and how they can be applied to the selected processing technique, namely MapReduce; big data storage and processing involving emerging technologies and platforms; operations over data that may be accelerated significantly, and potential practical applications. We will also briefly discuss useful mathematical models and methods that can efficiently be applied at different stages of big data processing, from the top level to hardware acceleration relying on divide-and-conquer procedures and different types of system decomposition.

For acceleration purposes we will talk mainly about the MapReduce processing technique and the relevant model for distributed computations that was based originally on Java programming language and contains two important steps. *Map* takes and divides data into subsets of individual elements that are of lower complexity, and are represented by tuples (key/values pairs). A simple example can be found in tutorial [4]. *Reduce* at the second step takes the outputs from the first step and builds a smaller set of tuples. We will consider the *MapReduce* processing technique from slightly different point of view and apply hardware acceleration at specific sub-steps.

As a simplified example, let us consider a group of students with the main objective being to find out the number of occurrences of each student's name. Similar problems appear in many practical applications requiring acquisition, analysis and filtering of large data sets [5]. In [6] a data mining problem is explained, with an analogy to a shopping card. A basket is the set of items purchased at one time. A frequent item is an item that often occurs in a database. A frequent set of items often occur together in the same basket. A researcher can request a particular support value and find the items which occur together in a basket either a maximum or minimum number of times within the database [6]. Similar problems arise to determine frequent queries on the Internet, customer transactions, credit card purchases, etc. These require very large volumes of data to be processed within a day [6]. As an example, the following list depicts the first names of students from part of a group in Aveiro University (Portugal): Tiago, Maria, Maria, Maria, Tiago, Ana, Beatriz, Ana, Tiago, Carolina, Maria, Beatriz. The name Maria is the most frequent (4 times), and the name Carolina is the least frequent (once). Figure 1 shows the operations that are executed in the MapReduce procedure.

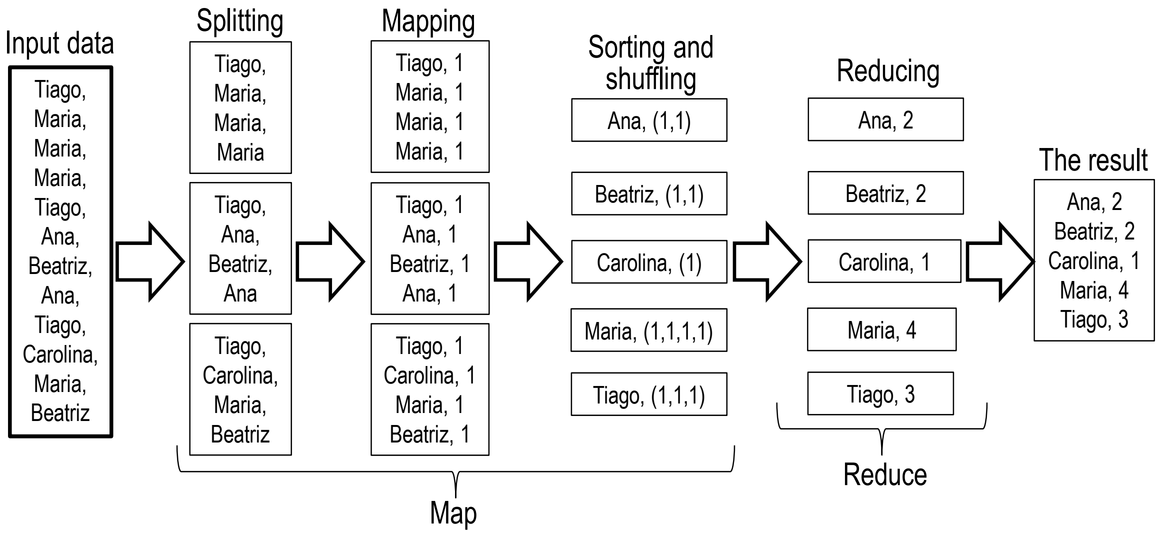

Input data
Tiago, Maria, Maria, Maria, Tiago, Ana, Beatriz, Ana, Tiago, Carolina, Maria, Beatriz
Splitting
Tiago, Maria, Maria, Maria
Tiago, Ana, Beatriz, Ana
Tiago, Carolina, Maria, Beatriz
Mapping
Tiago, 1 Maria, 1 Maria, 1 Maria, 1
Tiago, 1 Ana, 1 Beatriz, 1 Ana, 1
Tiago, 1 Carolina, 1 Maria, 1 Beatriz, 1
Sorting and shuffling
Ana, (1,1)
Beatriz, (1,1)
Carolina, (1)
Maria, (1,1,1,1)
Tiago, (1,1,1)
Reducing
Ana, 2
Beatriz, 2
Carolina, 1
Maria, 4
Tiago, 3
The result
Ana, 2 Beatriz, 2 Carolina, 1 Maria, 4 Tiago, 3
Map
Reduce

Figure *1.* – An example of the MapReduce procedure

Initially, the data set containing the names is divided into three subsets with four names in each. The number of objects (names) and the number of subsets have been chosen somehow. Initially, the number of appearances for each name is assigned to 1. The key is the name and the value is 1, for example «Tiago, 1». Then sorting and shuffling are done and all the tuples with the same name (key) are sent to the corresponding block. For example, the name Tiago from the first and the third mapping blocks are mapped to the fifth block created after sorting and shuffling: «Tiago, (1,1,1)» Counting the Hamming weight (the number of ones) gives the final output shown on the right-hand side of Fig. 1. As we can see, the operations applied are: extraction of non-repeated names; sorting; and counting the number of ones for each such name.

There are many hardware accelerators for executing the operations identified above that are proposed in [5] with all necessary details including hardware specifications in VHDL (Very high speed integrated circuits Hardware Description Language) and software models described in Java. In [7-9] many other useful and easily understandable software models are described in C/C++ and Java languages.

The remainder of this paper is organized in 6 sections. Section 2 reviews and discusses basic solutions for the development of data centers that provide support for managing big data. Section 3 presents a brief review of FPGA-based hardware accelerators for some selected important problems; these are sorting, the extraction of non-repeated names/values, discovering the most frequently occurring items, and Hamming weight computations. The importance of divide-and-conquer algorithms and system decomposition is emphasized. Section 4 is dedicated to emerging technologies and platforms for big data storage and processing. Section 5 overviews various operations over data. Section 6 demonstrates some feasible practical applications from the information system for electoral processes in Kazakhstan. The conclusion is given in section 7.

The material of the paper has been distributed between the authors as it is shown below:

– Sections 1, 3 and 5 have been prepared by M. Glesner (Germany), I. Horton (England), A. Keevallik (Estonia), I. Skliarova, and V. Sklyarov (Portugal) collaborating for more than 30 years. They have extensive experience in data processing based on divide-and-conquer algorithms, system decomposition at various levels, optimization applying discreet mathematical models, accelerations with FPGAs (Field-Programmable Gate Arrays) / PSoC (Programmable Systems-on-Chip), embedded systems, and computer-aided design techniques.

– Section 2 has been written by Y.G. Klyuyeva and V.V. Yavorskiy from Kazakhstan with long experience in big data management.

– Section 4 has been prepared by A. Adamov from Azerbaijan (sub-sections 4.1, 4.2, 4.4, 4.5, 4.6) and S.I. Gindin and A.D. Khomonenko from Russia (sub-section 4.3) with extensive experience in big data management. It should be noted that A. Adamov is the organizer of fourteen International IEEE conferences closely related to big data and analytics.

– Section 6 has been written by A.I. Buranbaeva (Kazakhstan) based on her practical experience.

Basic solutions for the development of data centers

One of the main goals of developing data processing centers (DPCs) is to provide information support for computer-based decisions for all or main activities of the relevant organization. The nature and scale of the data analysis tasks that need to be solved also determines the approaches to choosing architecture and designing a data center.

Several large companies provide solutions for data centers and leading positions are held by Oracle, Microsoft, IBM, and Teradata [10]. Oracle is a leader in the field of database management system (DBMS) software and supports all possible architecture options, including clusters, symmetrical multiprocessor systems, and over 80 variants of the operating environment, including IBM mainframes, DEC VAX, UNIX, Windows, and many other platforms. Offered solutions ranging from entry-level, to high-performance systems are scalable and secure, which is also critical for business applications [11]. Oracle also provides support for compatibility with older solutions (three quarters of Oracle customers have been working with Oracle Database for more than 10 years). The disadvantages of Oracle include the high cost and complexity of licensing, as well as problems associated with the release of updates. When designing a data center, Oracle offers a wide range of products, from using a certified configuration to a device that is ready to configure the data center. Exadata proprietary solutions are also available: Oracle Exadata X2–2 for data centers and mixed workloads, Oracle Exadata X2–8 for cloud solutions, and Oracle Exadata Storage Expansion Rack X2–2 for increasing data center capacity. The competitive capabilities of Oracle are determined by the following factors:

–there is a set of ready-made applications for the development of data centers, providing a complete life cycle;

–the company is one of the leaders in sales in the field of data analysis;

–achieving compatibility with products made by other companies.

Microsoft has also a strong position in the database market. The target is creating an instrumental and technological environment that would minimize the cost of developing data centers and making this process accessible to the mass user. The solutions offered by the company in the field of data warehousing concentrate on the development of OnLine Analytical Processing (OLAP) tools [12]. The offered solutions are SQL Server DBMS and the Azure SQL DATABASE cloud service. The company received the highest satisfaction from customers for meeting their needs, value for money, service, support, and overall experience. Microsoft has also increased competitiveness by launching the free Developer Edition of SQL Server and Database Migration Service tools for migrating SQL Server and Oracle databases to Azure SQL Database. SQL Server is used to work with small and medium-sized databases, as well as for large enterprise-wide databases. But, despite the strengths, many corporate customers still do not consider this DBMS suitable for mission-critical applications. According to the experience if the number of users exceeds 2000, then you need to switch to a higher-level DBMS, for example, from Oracle. In the data center market, Microsoft offers its Microsoft Data Warehouse and Fast Track Data Warehouse solutions to provide data centers for customers who do not need a massively parallel database management system. Microsoft released its own mass-parallel architecture data center device, SQL Server 2008 R2 Parallel Data Warehouse, in November 2010. This architecture is implemented in future versions of products [13].

IBM Corporation offers both standalone DBMS solutions and data center devices. The IBM Smart Analytics System (ISAS) family is currently available on the market, with IBM Data Center software that is InfoSphere Warehouse for Unix, Linux, Windows, and z / OS. There are thousands of database clients worldwide. It is easy to rich functionality of the solutions, including the cloud and hybrid capabilities that the company's products have, as well as the active use of popular open source solutions (Hadoop, Kafka, Parquet, Spark, etc.) and backup and restore functions in or from Swift and AWS S3. The core for data center is the DB2 family of object-relational DBMSs. The data manipulation language is SQL. The advantage of IBM solutions is manifested when both the operational data processing systems and the data center are managed by IBM software, that is, when the so-called closed standard solution is offered. However, IBM's revenue and market share in the operational database systems has been declining for several years. The DB2 database system loses to most competitors in terms of transaction processing speed and data loading. There are also difficulties with pricing and licensing.

Teradata works for more than 30 years in the data center market in combination with the established equipment and specialized database analytics software. The products include data mining tools, departmental data mining and enterprise solutions, as well as cloud and big data products. Aster Data has added new features to the Teradata product line (such as MapReduce, unstructured data, and graphical analysis).

Hence, there are several options for the implementation of data centers within the framework of a typical architecture. Let us consider the features of technological solutions for some of them.

Virtual data warehouse. The architecture provides a real-time access to live data through the middleware software. This solution is based on a metadata repository that describes data sources, procedures for their preliminary processing, and formats for presenting information to the end user. The disadvantages of such solution are intensive network traffic, a decrease in the performance of a non-existent system, and the threat of data integrity violation in the event of unsuccessful data center users.

Data Kiosks. The architecture is a lightweight version of the data center thematic focus. There are data kiosks associated with an integrated or unrelated (stand-alone) data center.

Global data warehouse. The architecture is a single source of integrated organization data.

Data warehouses with a multilevel (mainly three-tier) architecture, or corporate data centers. The architecture is a kind of global data center in which three levels are technologically implemented. At the first level, the organization’s corporate data center is located. At the second level, related topic-specific data kiosks based on multidimensional DBMS are supported. At the third level there are user client applications with data analysis tools installed.

Built-in (combined) data storage. The architecture is a data center that is organically integrated into a virtual enterprise (Enterprise Information Factory, EIF) or used as a component of analytical support of business functions.

Corporate Information Factory (CIF). It is a development of enterprise data warehouse (EDW) architecture involving coordinated data extraction from sources, loading them into a relational database with a structure in the third normal form, using the constructed data center to populate additional repositories of presentation data.

Data Warehouse Bus architecture. In this architecture, the data center is not a single physical repository (unlike CIF). This is a “virtual” component, representing a collection of data items, each of which has a star architecture.

United (federated) data center. In this architecture, there are a number of data center instances that operate on a semi-autonomous basis and, as a rule, they are organizationally or geographically dispersed, but can be considered and managed as one large entity.

Significant differences in software from different manufacturers are determined by the following factors:

–the used data model;

–the degree of coverage of the life cycle;

–built-in support for various architectures;

–data processing language capabilities.

Attention may be paid to the following main trends. Manufacturers offer complete solutions for creating data warehouses. Leading software companies in the field of design and development of information systems with databases launch their own programs for data storage systems and provide a full life cycle for the development and maintenance. They offer off-the-shelf embedded storage solutions significantly reducing the time for the design and development of data centers.

From the point of view of using software and hardware platforms, solutions in the field of creating decision support system (DSS) based on data warehouses can be divided into three classes:

–combination of finished products (solutions) of different companies without direct programming;

–using a complete closed chain of products (solutions) of one supplier company;

–using the contour of products (solutions) of one supplier's company with the addition to a closed chain of compatible products of third companies.

An example of a simple scalable solution can be based on the use of Crystal Enterprise and Crystal Reports (Business Objects) as end-user tools. The data center is implemented on DBMS Oracle, DB2, MS SQL Server or other similar components with open database connectivity (ODBC) interface or a direct access interface with Crystal Enterprise. Typically, classic data center architecture without data kiosks is used. For this decision, careful design of the data center structure and queries is of great importance. It is necessary to develop and create applications for data cleaning (or use the tools available from suppliers). The advantages of this solution are [14]:

– the amount of programming is minimized, since all stages are covered with ready-made box products;

– time is reduced for the development and creation of data centers (by eliminating the labor-intensive process of writing programs);

– the development time for a typical request is from 2 to 6 hours and for a typical report: 1-2 days;

– such a solution is good for creating prototypes of data centers, because almost all the necessary requests and reports are processed;

– an excellent environment for using non-core queries is created;

– this solution is also perfect for creating virtual data centers.

– The disadvantages of this solution are:

– developing complex cross-querying system may require long time;

– this solution is not suitable for complex analytical data processing, which needs the development of special applications for analysis.

A closed standard solution can be based on the use of a closed chain of products of one supplying company, for example Microsoft, Oracle, IBM. The advantages of this solution are:

–as a rule, all business areas are supported by ready-made services;

–the time of development and creation of a data center can be rigorously described and reasonably accurate;

–such a solution is good for creating data centers that are supposed to be used for a long time;

–such solutions are suitable for complex analytical data processing, which requires the development of special applications for analysis.

The disadvantages of closed standard solution are:

–high level cost of development and creation;

–requiring high-qualified staff to work with a set of products of the selected company.

Brief review of FPGA-based hardware accelerators

This section emphasizes FPGAs because this technology serves as a core of many modern systems-on-chip incorporating high-speed processing blocks, advanced interfacing capabilities (Peripheral Component Interconnect – PCI, Advanced eXtensible Interface – AXI, etc.), and GPUs (Graphics Processing Units) [15-18]. Besides, FPGAs themselves can be seen as very powerful architectural solutions for high-performance accelerators [5]. The related system design challenges in ubiquitous computing environments and open-source hardware initiatives have also been discussed in invited and keynote talks [19, 20].

A detailed review of FPGA-based hardware accelerators for the *MapReduc*e model is given in [21] with many details and examples of practical applications (see also section 4 below). FPGAs were invented by Xilinx in 1985 and they are the first truly programmable logic devices that could be configured after manufacturing. Nowadays they have quite impressive 35-years old history, showing evolution from relatively simple systems with a few tens of modest configurable logic blocks and programmable interconnects up to extremely complex chips combining the features of traditional FPGA with embedded multi-core processors and related peripherals, leading in this way to hardware Programmable Systems-on-Chip (PSoC). One of the most-recently unveiled reconfigurable accelerator platforms from Xilinx, Versal integrates FPGA technology with ARM CPU cores, Digital Signal Processing (DSP) and Artificial Intelligence (AI) processing engines and it is intended to be employed to process data-intensive workloads running datacenters [16-18]. Thus, they are appropriate for data processing. All necessary details about FPGAs are presented in [5, 22] and the book [22] has also been translated to Chinese in 2018.

Sections 1 and 4 list frequently used operations in the *MapReduce* model, namely sort, search, count, extraction of non-repeated names/values, discovering the most frequent items, Hamming weight computations and some others. One of the fastest sorters can be implemented in sorting networks, the majority of which use Batcher even-odd and bitonic mergers [23, 24]. A sorting network is a set of vertical lines composed of comparators that can swap data to change their positions in the input multi-item vector. The data propagate through the lines from left (from inputs) to right (to outputs) to produce the sorted multi-item vector on the outputs of the rightmost vertical line. Thus many-items data may be completely sorted in combinational circuits in such a way that input items are converted to sorted output items without applying clock cycles. There are also very efficient iterative solutions [25]. The book [5] describes effective hardware circuits including communication-time architectures allowing data to be completely sorted almost immediately as soon as a new item is received and, hence, minimizing communication overhead that is frequently pointed out as the main bottleneck in system performance. Thus, sorting illustrated in Fig. 1 may be implemented very fast.

The results of sorting may be further processed by the circuits proposed in [5]. Suppose there is a set of N sorted data items, which eventually includes repeated objects and we need the most frequently repeated item to be found. One possible solution for this problem is given in [5, pp. 98-102] (see Fig. 2 from [5]) where N–1 parallel comparators form a binary vector V. The most frequently repeated item can be discovered if we find the maximum number of consecutive ones in the vector and take the item from any input of the comparators that forms the sub-vector with the maximum number of consecutive ones.

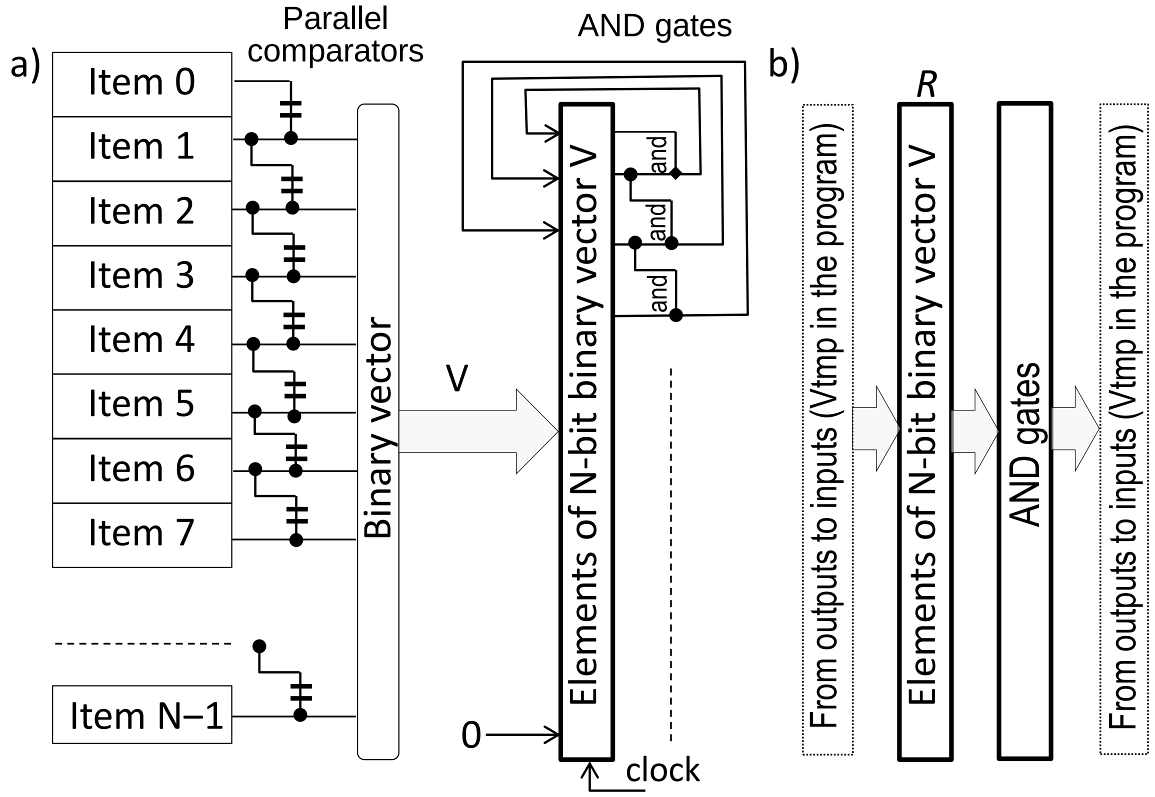

a)
Parallel comparators
Item 0
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item N−1
Binary vector
V
AND gates
and
and
and
Elements of N-bit binary vector V
0
clock
b)
R
From outputs to inputs (Vtmp in the program)
Elements of N-bit binary vector V
AND gates
From outputs to inputs (Vtmp in the program)

Figure *2*. – Architecture for the most frequent item computation in a given sorted set of data items (a), copying the vector from the outputs of AND gates to the register R (b)

Much like it was done in [5], the considered circuits may be slightly modified, which permits to discover the number of repetitions for all the items or to find only such items that are repeated more than κ times where κ is a given threshold. Thus, sorting and shuffling may completely be implemented in hardware. It is shown in [5, pp. 34-36] that the number of consecutive ones can be found in a very fast FPGA LUT (Look-Up Table)-based combinational circuits. To understand the functionality of this circuit a Java program modeling the respective behavior is given. Note that computing frequencies of integers in large data sets may be effectively done with the address-based technique [5], proposed in [26].

Other useful hardware designs permit the Hamming weight of a binary vector to be computed that is helpful for the reducing procedure (see Fig. 1). There are a number of very effective solutions that have been proposed for solving such problems [27-30]. Comparisons of hardware and software implementations are done in the shown above publications. Communication mechanisms between different modules described in software programs may be formally transformed to the relevant hardware circuits using the methods [31] relying on the model of hierarchical finite state machine [32].

Note that the described above methods are applicable for data sets with limited number of items. The operations over such sets are executed in hardware very fast significantly improving the relevant throughput comparing to software implementations. Clearly the proposed methods cannot be applied directly to large sets that are common for big data. The main idea is working with blocks of data with limited complexity in such a way that large data sets are decomposed on blocks and completing the applied operations (within the blocks) makes possible to further handle large blocks of data instead of individual items. A good example is merge sorting in which the preliminary sorted blocks are merged in concurrently executed computational units. Thus, items in the blocks are sorted in very-high speed hardware accelerators with subsequent merging of the sorted blocks in parallel software. Interfacing circuits undoubtedly reduce the performance. However, deep parallelization and optimization can be applied. Similar idea may be used for other types of data processing. Many examples can be found in [5, 22], which are supported by relevant hardware-level specifications and Java models. It is shown in [5] that such operations as Hamming weight computations may be completely implemented in on-chip (on-FPGA) solutions for hundreds of thousands bits with minimal delays from inputs to the results. Particular circuits and systems (used for the MapReduce model) have been discussed in [21] with examples. Note that open-source hardware may support operations over blocks [20], which is important for the future. The Gathering for Open Science Hardware (GOSH) is a diverse, global community working to enhance the sharing of publicly-available scientific technologies.

It should be noted that the techniques reviewed above can be used for complete on-chip solutions in embedded applications that (although are outside of the scope of big data) can be seen as very important for numerous practical needs [19]. Complete tasks (including software and hardware) may be solved on a single and relatively cheap microchips, such as [33], which provide very efficient interfaces for data exchange between processing units and programmable logic that have been evaluated in [34,35]. Other practical applications can be found in the scope of combinatorial optimization [36-38] based on mathematical models from [39, 40], such as transformations and operations over Boolean and ternary matrices. The major work in this area has been done by a corresponding member of Byelorussian Academy of science A.D. Zakrevskij. Three authors responsible for this section (A. Keevallik, I. Skliarova, V. Sklyarov) have been collaborating with A.D. Zakrevskij during many years, which is indicated in particular in the book [41]. An excellent paper [42] about scientific heritage of A.D. Zakrevskij was published by his colleagues in 2017. Many other practical applications [e.g. 3,43] may benefit from the proposed highly parallel hardware accelerators. The methods [5] permit LUTs contents to be generated from Java programs based on specification of the desired functionality in software. Thus, new fast hardware blocks can be designed easier because they may be carefully tested in software and then automatically retrieved and converted to hardware description language specifications. Information about network-based circuits in Russian can be found in [44].

The growing importance of hardware modules in different areas is emphasized in the keynote talk [20] where the following ideas are highlighted. In the contrary of open source software that is a well-known topic; open source hardware is a fairly new one. Indeed, as we approach the limits of available electronic systems in silicon, many specific designs in the past moved to generic standard chips. Best examples are electronic components like Arduino and Raspberry-Pi. In this domain we see an increased offer of high performance boards from companies like Intel. According to the executive summaries of the GOSH-community hardware forms a vital part of the scientific experimental process and the current supply chain limits access and impedes creativity and customization through high mark-ups and proprietary designs. Open source hardware enables designs to be shared, which often take advantage of modern digital fabrication techniques. Expanding the reach of this approach within academic research, citizen science and education has potential to increase access to experimental tools and ease their customization and reuse while lowering costs. In the DIY (Do It Yourself)-communities a growing number of people around the world are developing and using open source hardware in the context of the wider movement for Open Science that is a trend referred in proposals of the world-wide acting Open Science Hardware (OScH) community. However, in Europe such a movement is only slowly making progress. Thus, the status of the OScH approach has to be widely discussed to develop an open science hardware ecosystem which will play an important role for future smart system designs in different areas of the engineering community, including the basic directions of this conference. The referenced above reusable modules from [5, 22] can be seen as certain efforts done in such a field and targeted to high-performance data processing.

Big Data Storage and Processing: Emerging Technologies and Platforms

Preliminary remarks. The extremely fast grow of Internet Services, Web and Mobile Applications and advancements in the Pervasive, Ubiquity and Cloud Computing concepts have stimulated production of tremendous amounts of data partially available online (call metadata, texts, emails, social media updates, photos, videos, location, etc.). Even with the power of today's modern computers it is still big challenge for business and government organizations to manage, search, analyze, and visualize this vast amount of data turning it into actionable information and knowledge.

Just small part of this huge amount is structured (Databases, XML, logs) or semi-structured (web pages, email), over 90% of this information is unstructured, what means data does not have predefined structure and model. Generally, unstructured data is useless unless applying data mining and analysis techniques. At the same time, just in case if organizations are able to process and understand operational data, their efforts worth anything, otherwise they becomes useless. Data-Intensive computing, which is intended to address these problems, attracts great attention of research community over recent years yielding strong results. Data intensive computing framework is a complex system which includes hardware, software, communications, and Distributed File System (DFS) architecture.

Two key components of any data-intensive system are: Data Storage and Data Processing. So, which technologies, techniques, platforms, and tools are the best for Big Data storing and processing? How Big Data Era effects technological landscape? Why particular concepts and paradigms are better for Big Data? These and many other questions will be addressed in this research.

Big Data Problem. Moving 1 TB Data from one physical machine to another through network may take from 3 to several hours depending on specifications of platform and devices. Processing of 1 TB data may take from hours to forever depending on the Data format and how sophisticated the processing task is. It becomes clear that traditional platforms can’t cope with continuously increasing amounts of data. A single machine, even exceptionally powerful (supercomputer), cannot process or even store all the data.

The only solution is to use specially designed platform/architecture that distributes data across large cluster of physical machines. But, when we deal with huge amounts of data and many computers, the following questions can’t be ignored:

– How to split Data across many machines?

– How to minimize or avoid Data movement? This is because moving Data over network is expensive.

– Having many nodes/machines in any cluster increases failures. How to deal with failures?

Hadoop is open source common platform that combines two main tasks of any operating system: storing and processing data. Unlike to traditional systems, Hadoop accomplishes these tasks towards Big Data. The popularity of Hadoop increases day by day, because of simplicity, scalability and affordability that it provides for, thanks to its distributed architecture. Although, the Hadoop Core consists of two main components (Hadoop Distributed File System – HDFS and MapReduce) and has a limited functionality, thanks to many other components available in the Hadoop Ecosystem under Apache License, this platform can cover almost any demand to manage and process data regardless to its size and format [45].

Big Data Storage with Hadoop Distributed File System – HDFS. Data storage domain is in the center of the big data scientific studies and technological practices focused on machine learning, data mining and business intelligence. Storing large amounts of data allows many engineering problems to be efficiently solved such as: search and analyze the patterns, optimization for fastest access in business processes, etc.

HDFS is one of the most well-known examples of an open source distributed processing framework that manages data storage for big data applications. HDFS’s main advantage is the cost-efficient storage capacity for extremely large amounts of data and possibility to transfer data at high speed to user applications.

Typically organized in large clusters, user applications run on thousands of spread servers directly connected to the storage devices. Distributing data storage and computing resources between multiple servers allows scaling based on low-end hardware (if necessary) and creating an economically profitable system of any size. Hadoop clusters enable computing resources, storage capacity and input/output channel bandwidth to be adjusted by modifying the number of workstations in distributed network. Hadoop systems are very widely used to handle various forms of structured and unstructured data, known as Data Lakes [46], giving users more flexibility for collecting, processing, analyzing and managing data than relational databases and data warehouses provide.

Hadoop is the fundamental Big Data storage and processing technology. It is one of the most popular projects of the Apache Software Foundation. It is supported by a full stack variety of related technologies, such as:

– YARN (Yet Another Resource Negotiator) – the system for task planning and cluster management providing sharing, scaling, and reliability of distributed applications.

– Hadoop *MapReduce* – the platform for programming and performing distributed *MapReduce* [47,48] computing using a large number of nodes organized in a cluster.

– Apache Hive – the data warehouse software project built on top of Apache Hadoop for providing data query and analysis.

– Apache Spark – the open-source distributed general-purpose cluster-computing framework.

– Sqoop – the tool designed to efficiently transfer large amounts of data between Hadoop and structured DBMSs (for example, relational) in both directions.

The example of integration of multiple Big Data Processing Tools for Image Classification is considered in [48,49]. DataStore functionality built into MATLAB is used to develop and debug algorithms with a piece of data and then apply them on a distributed Hadoop cluster within the Hadoop MapReduce environment on the complete set of data.

The HDFS file system stores metadata and application data separately on a metadata server (NameNode) and application data servers (DataNode). All servers communicate with each other through protocols based on the transmission control protocol (TCP). Unlike the Lustre and parallel virtual file system (PVFS), the HDFS application data servers (DataNode) do not rely on such data protection mechanisms as redundant array of independent disks (RAID) to increase the reliability of data storage. On the contrary, similarly to the Google file system (GFS), the contents of a file is distributed and duplicated among many application data servers for its consistent storage. Ensuring the reliability of data storage, the HDFS strategy allows to take advantage of multiplying the bandwidth of the data transfer channel, and so when required, it becomes possible to carry out calculations in parallel with the delivery of the necessary data.

HDFS uses the classic UNIX tree structure of directories, user concept with read-write-execute operations of rights granting, and even a similar set of console commands. The data is divided into blocks (usually 64 MB or 128 MB), for each file the NameNode stores its path, a list of blocks with file data and their replicas.

The NameNode ensures that each block always has a given number of copies. It checks the fact that there are not enough copies or excessive copies of a block at the time of delivery of the block report from the DataNodes. In case the block has extra copies, the NameNode selects which copy to delete. The HDFS logic is designed to maintain the number of physical racks on which copies are stored, and to remove the copy from the DataNode on which the least amount of available disk space is available. The purpose of this policy is to balance the use of storage devices of data servers of application data without reducing the availability of blocks.

When the number of copies of a block becomes insufficient, the block enters the priority replication queue. A block with only one copy has the highest priority, and a block with several copies that make up more than two-thirds of the required replication volume has the lowest priority. A background program thread periodically scans the beginning of the replication queue to decide where to place the new block copies. The block replication process uses a simple policy for placing new copies of blocks. If one copy of the block is available, HDFS places the next copy on the server from another rack. In the case when the block has two available copies, if two existing copies are on the servers of one rack, the third copy is created on the server from the other rack; otherwise, the third copy is hosted on another server of the same rack used by the existing copy. In this case, the goal of the policy is to reduce the cost of resources for creating new copies.

User applications access the file system using the HDFS client, which is a dedicated library that exports the HDFS file system interface. Much like the most traditional file systems, HDFS supports reading, writing and deleting files, as well as creating and deleting directories. The user describes files and directories supplying paths in the namespace. The user application does not need to worry about the fact that metadata and file data from the file system are stored on different servers or that the blocks have many copies.

When the application reads from a file, the HDFS client first queries the metadata server for a list of application data servers that store copies of the data blocks of the required file. The list is sorted based on the distance to the client within the network topology. The client connects directly to the application data server and requests the transfer of the necessary block. When a client writes, it first requires the metadata server to select application data servers to store copies of the first block of the file. A client organizes a channel between several servers and sends data. When the first block is transmitted, the client requests the selection of the following application data servers for storing copies of the next block. A new channel is organized and the client sends data to the next block. The choice of application data servers for each block is likely to be different.

An important area of current research is the evaluation of operational efficiency of solutions and algorithms for processing big data. Given the complexity of the interactions architecture in the distributed systems, increasing the speed of data exchange in the file system is one of the main incentive targets to boost the overall Big Data implementation.

Examples of modeling research for assessing the efficiency of access to big data storage systems are considered in [50-53].

Big Data Parallel Processing with MapReduce. MapReduce is a distributed, large-scale data processing paradigm (also known as parallel programming concept) that was initially developed by Google [54]. The idea of MapReduce is very much similar to high order functions in functional programming languages. For example, Map and Fold (Reduce) functions in Haskell programming language.

Leveraging Big Data technologies, hundreds of SMEs and tech giants like Google, Amazon, Facebook, Twitter, etc. use Hadoop Ecosystem as a testing and production environment building their critical services on the top of this framework [55].

MapReduce is a programming concept designed to build algorithms for distributed computing (Figure 3).

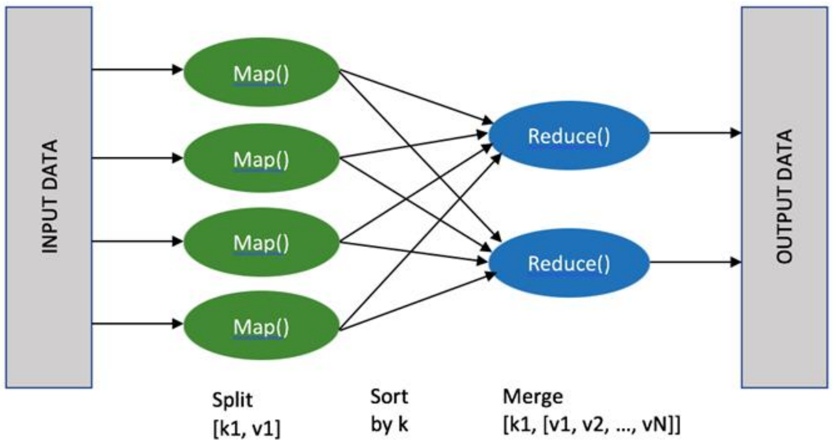


Figure *3.* – MapReduce Architecture

When someone develops an algorithm for distributed system that consists of many connected nodes, there appears a need to deal with many sophisticated issues not having direct relations to the problem intended to be solved. These issues are the followings:

- How to split a job to tasks to be distributed over computing nodes?
- What happens if any of nodes that was assigned with a task, fails at the mid of computation?
- How to distribute data/messages between nodes in the most effective way?
- How to minimize data flow between nodes of the distributed system?

And many others...

It is known that *MapReduce* is not equally well for any kind of distributed algorithms. Developers of *MapReduce* had a specific list of tasks (search, sort, count, etc.) for those this concept is tuned. The framework that implements *MapReduce* concept is optimized to accomplish these tasks in the most effective way, at the same time taking care about all low-level common issues we mentioned earlier. The problem with this concept is: in order to run specific task in the framework, it should be ensured that the task fits into frames of *MapReduce*. In other words, task must follow all prerequisites of *MapReduce* [56]. As it has been mentioned *MapReduce* is far away of being panacea for parallel programming tasks. It is designed for batch processing and is not efficient for the certain types of massive processing like:

- stream and Real-time data processing;
- iterative data processing;
- low-latency data processing.

Understanding MapReduce Lifecycle in detail. MapReduce is suitable for large problems that can be decomposed into sub-tasks not having dependency to each other. The introduced constraint enables almost infinite scalability. Depending on the scale, the problem (data) can be parallelized involving from several to thousands computing nodes. Besides, Hadoop provides fault-tolerance not just at the level of HDFS, but also at the level of MapReduce taking care of possible failure of nodes in each cluster. From programmer's point of view, it is quite easy to implement parallel programming on the top of MapReduce framework. Generally, a programmer deals with Map- and Reduce-functions, and framework itself takes care about all possible system-level operations, including fault-tolerance, data distribution, etc. Any task that is intended to be run using MapReduce framework has to be divided into three essential functions [57] (see Fig. 4): 1) Map; 2) Sort/Shuffle/Merge; 3) Reduce.

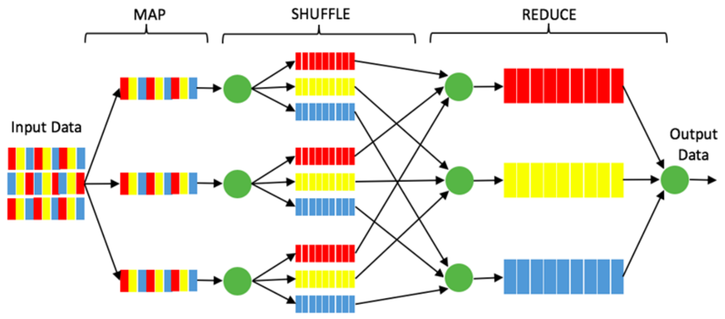


Figure *4.* – Phases of MapReduce Lifecycle

The *Map*-function turns the input data that usually comes as blocks from HDFS into key-value pairs and applies specific functions to each element of input data. The *Map* function can be compared to mapping in functional programming languages (for example Haskell) map func [e1, *e2*, *e3*, ..., *eN]*, where function func is applied to each element of the list generating new list. The second task (Sort/Shuffle/Merge) sorts (or shuffles according to predefined conditions) pairs from all Mappers preparing data for the next task. The Reduce-function extracts each unique key and associates it with all values (similar to group by key operation) prepared by Map-function. It is important to notice that, while MapReduce runs each of Map-Shuffle-Reduce functions, it follows to requirement of Data Locality. Doing so, *Map*-function is executed on particular node where one of 3 data-block replicas (default replication factor in HDFS) resides avoiding any need for data transfer. In the same way, Shuffle- and Reduce-functions are executed on the node where Map-function stores its output. As a result, computations (in the form of software code) that implement those functions are transferred to where data is, while data itself remain immovable [58].

There are three categories of data locality with different level:

– Node-level locality - processing takes place on the same node where data resides;

– Rack-level locality - processing takes place on the node different from that where data resides, but on the node that belongs to the same rack;

– Cluster-level locality - processing takes place on the node located on different rack.

The fundamental features of MapReduce that made this paradigm de-facto standard for large-scale data processing are the followings:

– Move computation to the data rather than transferring data. Transferring data, especially huge amounts of data, over network is timely and costly process. For example, to move 1 TB file over 1 GB Ethernet network will take between 3 to 6 hours depending on many parameters of computers and network (performance of hard disks, performance of computers, network traffic usage, etc.).

– MapReduce deals with data stored in the Hadoop's native file system HDFS. HDFS with unique features of high availability, fault-tolerance, distributed storage and replication is available for all phases of MapReduce processing. The input, intermediate and output data are stored on distributed HDFS.

– MapReduce manages all resources available in a cluster. It is particularly true taking into account MapReduce transformation into YARN (Yet Another Resource Negotiator). YARN does not just manage resources (memory and processing power) among applications and jobs, but also responsible for provision of fault-tolerance, creating new containers, re-assigning jobs to other machines in case of failure.

Logical Model of MapReduce. To understand the logical model of MapReduce framework, let's look to the following example, which can be considered as starting point for implementation of Natural Language Processing (Text Analytics) application on the top of MapReduce. Let *W* be large textual dataset of length *n*. The i^{th} word in the sequence *W* is depicted as ω_i. Since MapReduce expects the input to be a list of key-value pairs, the W is presented as the list of n pairs $< i, \omega_i >$. The MapReduce lifecycle can be introduced as following process:

– Each word in textual dataset is depicted as the key, and the position of word as the value. So, the *Map*-function can be defined as $\mu_i(< i; \omega_i >) \ = < \omega_i; i >$.

– After grouping by key (particular word) as an output of *Map*-function, *Reduce*-function will receive the unique key (word) and the list of all positions of this word appears at. So, the *Reduce*-function can be defined as $\mathfrak{r}_i(< \omega_i; \{\rho_1, \rho_2, \dots, \rho_m\} >)$.

– All pairs with the same key will be submitted to the same reducer (see Fig. 3) [59]:

$$\mathfrak{r}(< \omega_i; \{\rho_1, \rho_2, \dots, \rho_m\} >) =< \omega_i; \sum_i \rho_i >$$

Operations over Data

There are many frequently executed operations over data and just a part of them from [5] is listed below:

– Retrieving from a given set items with the maximum and/or minimum values.

– Retrieving from a given set items with the maximum and/or minimum values of given parameters, such as the Hamming weight (HW) of lines/column of binary/ternary matrices, the maximum and/or minimum difference in the HWs in a given set of binary/ternary vectors, etc.

– Finding the most repeated items.

– Testing if in a given data set there are no repeated items, i.e. all values of the items are unique.

– Ordering repeated items that satisfy some predefined conditions, for example they must be repeated not less than a given threshold.

– Decomposing a given set in intervals $0,\dots,N_{sets-1}$ and solving the listed above tasks in each separate interval.

– Finding extreme values in the set of intervals, for example, retrieving all minimum values in each interval and finding the maximum value among the retrieved minimum values (finding the greatest minimum); similarly the smallest maximum can be requested to be found.

– Extracting sorted maximum/minimum subsets from a given set.

– Filtering data, i.e. extracting subsets with values that are within the defined limits or satisfy some characteristics.

– Dividing data items into intervals or subsets $\Theta_0,\dots,\Theta_{E-1}$ and finding the minimum/maximum/average values in each subset, sorting each subset, or solving some other problems partially described in the next point.

– Operations with the subsets $\Theta_0,\dots,\Theta_{E-1}$ such as union $\Theta_i \cup \Theta_j$ $(x : x \in \Theta_i$ or $x \in \Theta_j)$, intersection $\Theta_i \cap \Theta_j$ $(x : x \in \Theta_i$ and $x \in \Theta_j)$, difference $\Theta_i - \Theta_j$ $(x : x \in \Theta_i$ and $x \notin \Theta_j)$. Some other operations may also be considered, for example, removing from a given subset all repeated or indicated items, etc.

– Locating and ordering repeating relationships between different objects, for example ordering the most/less frequent items or items satisfying some other criteria.

– Removing all duplicated items from a given set.

– Multi-targeted sorting, i.e. ordering the same set by different criteria.

– Solving the problems indicated above for matrices (for rows/columns of matrices).

The above operations may be needed when big data are processed.

Practical Application

The described below practical application is taken from an experience in Kazakhstan. When designing an information system for electoral processes, one must take into account the need to process a large amount of streaming data at all stages of voting [60]. In this regard, it is important to consider scalable, flexible and fault-tolerant technologies. Thus, the use of micro-service architecture for electronic voting systems has become popular. The concept of «micro-service architecture of information systems» has gained popularity over the past few years, as a description of the method of designing an information system in the form of a set of independently deployable services. While there is no exact description of this architectural style, there is a certain common set of characteristics: organization of services around business needs, automatic deployment, transfer of logic from the message bus to receivers (endpoints) and decentralized control over languages and data.

The main reason for using services instead of libraries is independent deployment. If you are developing an application consisting of several libraries working in one process, any change in these libraries leads to redesign of the entire application. But if the application is divided into several services, then the changes affecting any of them will require a reconstruction of only the changed service.

If the micro-service of one of the modules stops responding as a result of an accident, its clients should be immediately redirected to the backup one. So-called message queues are often used to control the flow of requests. The use of a message queuing system ensures reliability and the necessary performance in data collection and processing. A message queue is a form of an asynchronous communication between modules (services) of a system. Messages are queued until they are processed and deleted. Each message is processed only once and only by one consumer. Message queues can be used for the separation complex processes, to buffer or organize batch handling, and to smooth out peak loads. A queue can be used by multiple sources and recipients, but each message is processed by one recipient only once. For this reason, this type of messaging is often referred to as one-to-one or direct.

A micro-service with message queues for the electronic voting system can use the data collection module. Figure 5 illustrates the operation algorithm of such a data acquisition micro-service module.

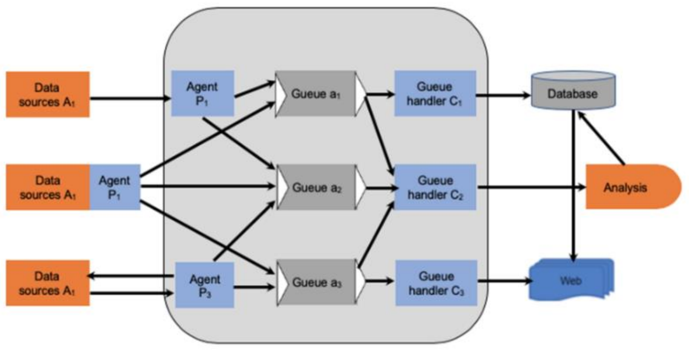


Figure 5. – The algorithm of the micro-service data acquisition module: A - data sources; P - agents to set the data in the queue; a - message queues; C - queue handlers

The module is a set of distributed agents that is responsible for the direct collection of data, the initial processing of data, and bringing the data to the standard form of the internal ecosystem of the electronic voting platform. The source of information for the data collection module, depending on the functionality and type of user, can be client personal computers, tablets or smartphones of voters (A1, A3), voting terminals and electronic ballot boxes (A2). Input agents (P1, P3) receive appropriate data streams or take portion from data sources (P2). Data queues and output queue processing agents are formed using special algorithms that reflect the specifics of information processes ensuring election confidentiality. The received big data is transferred to the next level of the system to the corresponding modules. The presence of many agents solves the problems of various data formats, because it allows for initial data processing the computational load on the analysis unit to be reduced. The considered module of the electronic voting system can use Apache Kafka platform developed by LinkedIn Company subsequently transferred to Apache Foundation [61]. Apache Kafka is a disseminated streaming platform, as well as a very fast distributed message broker that allows the required functionality to be implemented. Kafka agents (producer and consumer) can be implemented in various modern programming languages, such as Python, Java, Scala. The Kafka architecture is based on the Publisher-Subscriber concept.

Let us compare the functionality of Kafka with its analogues. The functionality is limited to only one pattern, thereby being inferior in functionality to many other platforms, such as the RabbitMQ platform [62]. However, other platforms delete messages after successful delivery, while Kafka stores them for any required period, by default a week, thus increasing message delivery guarantees in case of failure of subsequent dependent modules, which is very important for the electronic voting system. Also, the requirements for the network bandwidth and equipment resources on which Kafka will be launched are lower than those of the popular ActiveMQ and RabbitMQ message brokers, as well as the bandwidth figures: 20 Kbit / s messages per second for Kafka versus 100 Kbit / s for RabbitMQ. In terms of clustering, Kafka has an advantage over analogues ease of customization and scalability. Since one Kafka broker is a cluster, to add a second instance of Kafka is enough to inform the first instance about the presence of the second. Kafka saves messages to disk, which allows synchronizing end layouts between cluster nodes instead of data replication, while maintaining the logic of processing topics. Also, the output throughput grows exponentially, since a single topic can be processed by several brokers [61].

The joint use of this solution based on the micro-service architecture on the Kafka platform and hardware accelerators based on FPGAs with the MapReduce model [20] (see also section 3 above) can be a breakthrough option in the automation of elective processes.

Conclusion

The paper combines expertize of a large group of authors actively working in different areas that are important for big data processing. Various sections of this paper briefly describe their results. Distribution of the material between the authors is shown at the end of section 1. Additional information may be taken from publications of the authors shown in the list of references given below.

References

[1]. High Performance Computing for Big Data. Methodologies and Applications. Edited by Chao Wang. CLR Press by Taylor & Francis Group, 2018.

[2]. C.L.P. Chen and C.Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: a survey on big data", Information Sciences, vol. 275, 2014, Pp. 314-347.

[3]. B. Parhami, "Computer architecture for big data", in Encyclopedia of Big Data Technologies, S. Sakr, A. Zomaya (eds.), Springer, 2018.

[4]. MapReduce Tutorial – Fundamentals of MapReduce with MapReduce Example. Available at https://www.edureka.co/blog/mapreduce-tutorial.

[5]. I.Skliarova, V.Sklyarov. FPGA-based Hardware Accelerators. Springer, Switzerland, 2019, 245 p.

[6]. Baker Z.K., Prasanna V.K. An Architecture for Efficient Hardware Data Mining using Reconfigurable Computing Systems. In: Proceedings of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines – FCCM'06, Napa, USA, April 2006, Pp. 67-75.

[7]. Ivor Horton, Beginning C - From Novice to Professional Apress, 2018, 614 p.

[8]. Ivor Horton, Using the C++ Standard Template Libraries, Apress, 2015, 489 p.

[9]. Ivor Horton, Beginning Java, Willey Publishing, 2011, 1112 p.

[10]. Калимолдаев М.Н., Утепбергенов И.Т., Яворский В.В., Ахмедиярова А.Т., Клюева Е.Г. Организация хранилищ данных для Смарт систем городского общественного транспорта. Алматы: Институт информационных и вычислительных технологий КН МОН РК, 2019, 119 с.

[11]. Bruce Nelson, Saurabh Gupta and others. Oracle Big Data Handbook. Oracle Press, 2013, 464 p.

[12]. Brian Mitchell, Christopher Price, Dan Clark, John Welch, James Rowland-Jones, Adam Jorgensen. Microsoft Big Data Solutions. Wiley, 2014, 408 p.

[13]. Деян Сарка, Матия Лах, Грега Йеркич, Microsoft SQL Server 2012. Реализация хранилищ данных. Учебный курс Microsoft Русская редакция, 2012, 816 с.

[14]. Су Кеннет, Анналин Ын. Теоретический минимум по Big Data. Всё что нужно знать о больших данных. Питер, 2019, 208 с.

[15]. M. Santarini, "Xilinx 16nm UltraScale+ devices yield 2-5X performance/watt advantage", XCell Journal, issue 90, 2015, Pp. 8-15.

[16]. Xilinx Press Releases (2018) Xilinx Unveils Versal: The First in a New Category of Platforms Delivering Rapid Innovation with Software Programmability and Scalable AI Inference. https://www.xilinx.com/news/press/2018/xilinx-unveils-versal-the-first-in-a-new-category-of-platforms-delivering-rapid-innovation-with-software-programmability-and-scalable-ai-inference.html.

[17]. Xilinx Inc. (2018) Versal: The First Adaptive Compute Acceleration Platform (ACAP). White Paper: Versal ACAPs. https://www.xilinx.com/support/documentation/white_papers/wp505-versal-acap.pdf. Xilinx Inc. (2018) Xilinx AI Engines and Their Applications. White Paper: AI Engines. https://www.xilinx.com/support/documentation/white_papers/wp506-ai-engine.pdf.

[18]. Manfred Glesner, Thomas Hollstein, Tudor Murgan: System Design Challenges in Ubiquitous Computing Environments, in: Proc. of the Intl. Conf. on Microelectronics (ICM), S.11-14, 2004. Invited Talk.

[19]. Manfred Glesner, "The rebirth of hardware: Open-source hardware initiatives for IoT- and MINT-based developments". Proceeding of the 2nd International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC 2017), July 12-14, 2017, Madrid, Spain (http://www.cei.upm.es/recosoc17/keynotes.html).

[20]. V. Sklyarov, I. Skliarova, I. Uterbergenov, A. Akhmediyarova, "High-Performance Information Processing in Distributed Computing Systems", International Journal of Innovative Computing, Information and Control, vol. 15, no. 1. 2019, Pp. 321-335.

[21]. Sklyarov V, Skliarova I, Barkalov A, Titarenko L. Synthesis and Optimization of FPGA-Based Systems, Springer, Switzerland, 2014, 432 p.

[22]. Batcher K.E. Sorting networks and their applications. In: Proceedings of AFIPS Spring Joint Computer Conf., USA, 1968.

[23]. Aj-Haj Baddar S.W., Batcher K.E. Designing Sorting Networks. A New Paradigm. Springer, 2011, 132 p.

[24]. Sklyarov V., Skliarova I., "High-performance implementation of regular and easily scalable sorting networks on an FPGA", Microprocessors and Microsystems, vol. 38, no. 5, July 2014, Pp. 470-484.

[25]. Sklyarov V, Skliarova I, Mihhailov D, Sudnitson A. Implementation in FPGA of Address-based Data Sorting. In: Proceedings of the 21st International Conference on Field Programmable Logic and Applications, Crete, Greece, 2011, Pp. 405-410.

[26]. Sklyarov V, Skliarova I. Design and implementation of counting networks. Computing, 2015, 97(6):557-577.

[27]. Sklyarov V, Skliarova I. Digital Hamming weight and distance analyzers for binary vectors and matrices. International Journal of Innovative Computing, Information and Control, 2013, 9(12):4825-4849.

[28]. Sklyarov V, Skliarova I, Silva J. On-Chip Reconfigurable Hardware Accelerators for Popcount Computations. International Journal of Reconfigurable Computing, 2016:8972065.

[29]. Sklyarov V, Skliarova I. Multi-core DSP-based Vector Set Bits Counters/Comparators. Journal of Signal Processing Systems, 2015, 80(3):309-322.

[30]. Sklyarov V, Skliarova I. Hardware Implementations of Software Programs Based on HFSM Models. Computers & Electrical Engineering, 2013, 39(7):2145-2160.

[31]. V.Sklyarov. Hierarchical Finite-State Machines and Their Use for Digital Control. IEEE Transactions on VLSI Systems, 1999, Vol. 7, No 2, Pp. 222-228.

[32]. Xilinx Inc. (2018) Zynq-7000 All Programmable SoC Technical Reference Manual. https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf..

[33]. Silva J, Sklyarov V, Skliarova I. Comparison of On-chip Communications in Zynq-7000 All Programmable Systems-on-Chip. IEEE Embedded Systems Letters, 2015, 7(1):31-34.

[34]. Sklyarov V, Skliarova I, Silva J, Sudnitson A. Analysis and Comparison of Attainable Hardware Acceleration in All Programmable Systems-on-Chip. In: Proceedings of the Euromicro Conference on Digital System Design - Euromicro DSD'2015, Madeira, Portugal, August, 2015, Pp. 345-352.

[35]. Skliarova I, Ferrari A.B. Reconfigurable Hardware SAT Solvers: A Survey of Systems. IEEE Transactions on Computers, 2004, 53(11):1449-1461.

[36]. Skliarova I, Ferrari A.B. A Software/Reconfigurable Hardware SAT Solver. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2004, 12(4):408-419.

[37]. Skliarova I, Ferrari A.B. The Design and Implementation of a Reconfigurable Processor for Problems of Combinatorial Computation. Journal of Systems Architecture. Special Issue on Reconfigurable Systems, 2003, 49(4-6):211-226.

[38]. A. Zakrevskij, Y. Pottosin, L. Cheremisinova. Combinatorial Algorithms of Discrete Mathematics. Edited by Andres Keevallik. Tallinn, TUT Press, 2008, 193 p.

[39]. A.Zakrevskij, Y.Pottosin, L.Cheremisinova. Optimization in Boolean Space. Edited by Andres Keevallik. Tallinn, TUT Press, 2009, 241 p.

[40]. А.Д. Закревский. В Томском университете, Минск, 2014, 224 с.

[41]. П.Н.Бибило, Ю.В.Поттосин, Л.Д.Черемисинова. О научном наследии члена-корреспондента А.Д.Закревского. Информатика, N 1 (январь-март), 2017, стр. 112-124.

[42]. Chee C.H., Jaafar J, Aziz I.A., Hasan M.H., Yeoh W. Algorithms for frequent itemset mining: a literature review. Artificial Intelligence Review, 2018.

[43]. Скляров В.А., Склярова Ю.В., "Обработка данных в программно-аппаратных системах логического управления на основе поисковых сетей", Автоматика и Телемеханика, no. 1, 2017, Pp. 121-136.

[44]. Abzetdin Adamov, Large-scale Data Modelling in Hive and Distributed Query Processing using MapReduce and Tez, DiVAI 2018 - Distance Learning in Applied Informatics, 02 - 04 October, 2018, Štúrovo, Slovakia.

[45]. Walker, Coral; Alrehamy, Hassan. "Personal Data Lake with Data Gravity Pull". IEEE Fifth International Conference on Big Data and Cloud Computing, 2015) Pp. 160–167.

[46]. Ghazi M.R., Gangodkar D. Hadoop, MapReduce and HDFS: A Developers Perspective. Procedia Computer Science. 2015 Jan 1;48:45-50.

[47]. Paramonov I.Yu., Smagin V.A., Kosykh N.E., Khomonenko A.D. Methods and models for the study of complex systems and big data processing. Ed. V.A.Smagin and A.D. Khomonenko. St. Petersburg Publishing House "Lan", 2020. 236 p.

[48]. Nikita E. Kosykh, Anatoly D. Khomonenko, Alexander P. Bochkov, Anatoly V. Kikot. Integration of Big Data Processing Tools and Neural Networks for Image Classification. Proceedings of Models and Methods of Information Systems Research Workshop in the frame of the Betancourt International Engineering Forum. St. Petersburg, Russian Federation, Dec. 4-5, 2019. CEUR Workshop Proceedings. Vol-2556, Pp. 52-58.

[49]. A.D. Khomonenko, A.G. Basyrov, V.P. Bubnov [et al.]. Models and methods of research of information systems. Edited by A. D. Khomonenko. St. Petersburg: Publishing House "Doe". Russia. 2019. - 204 p.

[50]. A.D.Khomonenko; S.I.Gindin; Khalil Maad Modher. A cloud computing model using multi-channel queuing system with cooling. XIX IEEE International Conference on Soft Computing and Measurements (SCM). 2016, Pp 103 - 106.

[51]. V.A.Lokhvitskii, A.D.Khomonenko and M.A.Bol'shakov, On the Construction of a Cybervisor for the Intelligent Monitoring and Control of Data Centers Automatic Control and Computer Sciences, 2019, Vol. 53, No. 8, pp. 870–873. © Allerton Press, Inc., 2019. Russian Text © The Author(s), 2019, published in Problemy Informatsionnoi Bezopasnosti, Komp'yuternye Sistemy.

[52]. A.D.Khomonenko, S.I.Gindin. Performance evaluation of cloud computing accounting for expenses on information security. 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT). 18-22 April 2016, Pp. 100-105.

[53]. Jeffrey Dean and Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, Google Inc.

[54]. Apache Wiki, Powered by Apache Hadoop, https://cwiki.apache.org/confluence/ display/HADOOP2/PoweredBy.

[55]. Bala, M., Boussaid, O., & Alimazighi, Z. A Fine-Grained Distribution Approach for ETL Processes in Big Data Environments. Data & Knowledge Engineering, 2017, 111, Pp. 114–136.

[56]. Sterling, T., Anderson, M., & Brodowicz, M. MapReduce. High Performance Computing, 2018, Pp. 579–589.

[57]. Lee, S., Jo, J.-Y., & Kim, Y. Hadoop Performance Analysis Model with Deep Data Locality. Information, 2019, 10(7), 222 p.

[58]. Howard Jeffrey Karloff, Siddharth Suri, Sergei Vassilvitskii, A Model of Computation for MapReduce, SODA '10: Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms, January 2010, Pp. 938–948.

[59]. I.T. Utepbergenov, A.I. Buranbaeva. Анализ международного опыта применения информационно-коммуникационной технологоо в выборных процессах. Almaty, 2018, 78 p.

[60]. Humphrey P. Understanding When to use RabbitMQ or Apache Kafka: https://content.pivotal.io/blog/understanding-when-to-use-rabbitmq-or-apache-kafka.pdf.

[61]. RabbitMQ vs. Kafka vs. ActiveMQ Technoblog StackShare.io. https://stackshare.io/stackups/ activemq-vs-kafka-vs-rabbitmq.pdf.