DOI: 10.14529/jsfi200103

Supercomputing Technologies as Drive for Development of Enterprise Information Systems and Digital Economy

Oleg V. Loginovsky[1]***, Alexander L. Shestakov***[1]***, Alexander A. Shinkarev***[1]

The article presents an analysis of approaches to the development of enterprise information systems that are in use today. One of the major trends that predetermines the agenda of information technology is the focus on parallel computing of large volumes of data using supercomputing technologies. The article considers the resulting ubiquitous move to distributed patterns of building enterprise information systems and avoiding monolithic architectures. The emphasis is placed on the importance of such fundamental characteristics of enterprise information systems as reliability, scalability, and maintainability. The article justifies the importance of machine learning in the context of effective big data analysis and competitive gain for business, vital for both maintaining a leading position in the market and surviving in conditions of global instability and digitalization of economy. Transition from storing the current state of a enterprise information system to storing a full log and history of all changes in the event stream is proposed as an instrument of achieving linearization of the data stream for subsequent parallel computing. There is a new view that is being shaped of specialists at the intersection of engineering and analytical disciplines, who would be able to effectively develop scalable systems and algorithms for data processing and integration of its results into company business processes.

Keywords: enterprise information systems, parallel computing, supercomputing technologies, big data, machine learning, scalability, event stream, analysis, digital economy.

Introduction

Today, enterprise information systems experience what can be called a rebirth. Only yesterday monolithic applications were considered the basis of software development, while today unscalable development patterns from the "past" are falling into oblivion and giving way to microservice architectures, which in turn support the further efficient digitalization of the economy.

However, scalability, fault tolerance and possibility to carry out parallel computing across a cluster of computers come at a price of a significant increase in the systems complexity. Deployment, trouble shooting and support of such complex systems are the tasks far from trivial, requiring specialists of higher technical level compared to the development of applications that do not require fault tolerance, process small volumes of data and run on a single machine, namely, on the end user's computer.

For many, microservice patterns for building information systems today is primarily a matter of fashion. However, the need to move to flexible, loosely coupled and scalable systems did not come out of nowhere. The volume and speed of data generation in recent years, as well as the predicted growth of this trend cast doubt on the relevance of the entire data processing on one device. Nowadays, it is impossible to count on the successful processing of the so-called big data without a cluster of computing nodes, parallel algorithms, or involving supercomputer processing power. An important driver for the spread of parallel computing in enterprise applications, in addition to accumulating huge volumes of raw data, is the popularization of data science and machine learning. Business is much more aware of the potential of this field than it was 10 years ago. It allows one to see achievable goals for data analysis, invest money in computing resources and specialists of a new school – data engineers.

[1]South Ural State University, Chelyabinsk, Russian Federation

When developing enterprise applications based on parallel computing, it is more important than ever to pay special attention to such fundamental characteristics as reliability, scalability, and maintainability. There is a growth in complexity with moving to scalable development patterns, which means that one needs to be more scrupulous about the quality of the code, testing, fault tolerance, breaking tasks into subtasks, and linearizing command streams.

Besides developing approaches to parallel enterprise computing, databases also gained much impetus. There is also a move to scalable data warehouses and shift away from distributed transactions for the sake of achieving scalability of computing. Thus, distribution and scalability are fundamental characteristics of modern technologies of enterprise software development.

In machine learning, like in other spheres, there is a move from data processing on a scientist's personal computer to calculations and receiving results from several computing nodes. This is again explained by the volume of data that needs to be processed, as well as by the extent to which a granular computational problem can be broken down into unrelated subtasks.

For a more comprehensive description of the current situation and emerging trends in the development of modern enterprise information systems based on parallel computing, let us consider big data in more detail, as the main reason of a quantum leap in the field of distributed computing and related disciplines.

1. Big Data

1.1. General Characteristics

The concept of big data has in a sense become too common lately, and its meaning is now quite indistinct.

Here is one of the existing definitions of big data, which is quite generalized, like the term that it describes.

Big data is a scientific and practical field associated with the development and application of methods and tools for operating large volumes of unstructured data [4].

Such an understanding of big data as a phenomenon caused by the development of modern information technologies raises a number of global issues. The first issue is defining the criteria for categorizing data as big. The second issue is the assignment of only unstructured data (schemaless data) to the category in question at this stage of technology, when NoSQL and relational databases, as well as approaches to the storage and indexing, are gradually overlapping [30]. We list the well-known characteristics of big data [7]:

- Volume – How much data is there?
- Variety – How diverse are different types of data?
- Velocity – At what speed is new data generated?
- Veracity – How accurate is the data?

There are several additional characteristics, such as viability, value, variability, and visualization [22].

Here is the list of typical sources of big data [35]:

- Internet of things.
- Social networks.
- Telecommunication satellites.
- Internet stores.
- Internet encyclopedias.

- Various debug logs.

Surely, in addition to the listed above, there are many other sources of big data, and their number will continue to grow.

We list the main types of generated big data [7]:

- Structured – data possessing a specific schema and a fixed set of attributes.
- Unstructured – data without a permanent structure, for example various text documents.
- Natural language – a special kind of unstructured data that includes all texts in all languages of the world.
- Machine-generated – created by a computer, application, or other machine without human intervention.
- Graph-based – the natural representation of which is a graph modeling pairwise relations between objects.
- Streaming – data create in response to the occurrence of an event.
- Audio, video, and images.

Having identified the sources of the big data generators and the types of information they contain, let us consider the data science working process. Here is a description of the process systematizing the work of the analyst [7]:

- Setting the research goal – what you are going to research, how the company benefits from that, what data and resources you need.
- Retrieving data – checking the existence of, quality, and access to the data.
- Data preparation – enhancing the quality and consistency of the data, its normalization and removal of invalid entries.
- Data exploration – analyzing how variables interact with each other, a deeper understanding of the nature of phenomena under study.
- Data modeling – an iterative process of building a model to answer the research question using the insights about the data you found in the previous steps.
- Presentation and automation – using a suitable way to present the work results, automating the execution of the process to update results in case of new source data.

The presented typical process helps to systematically address problems with data that can fit in the memory of a single computer, and answer questions posed to a much larger amount of data, the processing of which may require a whole group of servers (cluster). Scientists and programmers spend considerable amount of time solving the problems of distributed processing of large data sets and teaching models on them.

Bearing in mind the volume of data generated in the world, the importance of solving the problem of efficient distributed data processing cannot be underestimated. And this volume is truly impressive. Google handles 5.5 billion searches per day [23], the number of Internet-connected devices according to various estimates could reach 22–50 billion by 2020 [35] and 75 billion by 2025 [39], Instagram users create 95 million posts per day [8].

Yet, data volume alone cannot give a business a competitive advantage. Only correctly putting questions to this data, as well as quickly making the most of the answers and restructuring your business, one can justify the overhead costs and the ever-increasing complexity of information systems that big data brings. In order to effectively work with data, analysts and stakeholders need tools, and it is vital that they fit for solving existing problems.

1.2. Big Data Tools

Numerous libraries that hide the complexity of the models behind their software interface is one of the factors making the methods of working with big data more accessible to a wide range of specialists.

On the one hand, this has a positive influence on the speed of the introduction and dissemination of new approaches to business and their integration into a greater number of key business processes. On the other hand, the specialists who only yesterday developed document management systems, by way of example, have a gap between the theory and practice of using machine learning models. This may adversely affect the final result of their work when standard models that work "out of the box" are not enough, and they have to be adjusted or even cascaded with other models.

Therefore, to develop corporate IT infrastructure in the direction of data mining and automation of management decision-making support, the involved specialists need to study the root technologies and the mathematical apparatus that underlie any library of machine learning models. It will not be possible to solve problems on big data without mastering the patterns of parallel data processing, distributed storage, and scalable multi-threaded algorithms. MapReduce, proposed by Google and already considered a classic pattern, can be a good starting point, although it is not so common in Google anymore [37].

The above mentioned "fundamentals" should become the focus of attention when training new personnel in the future, because only this way makes possible the necessary breakthrough growth of automation and informatization of Russian companies at the level of Western competitors and higher.

Today, business is pushing technology development towards big data methods and systems.

There are various driving forces for such development [26]:

- Businesses need to be agile and respond to new market insights by quick and cheap hypotheses testing and short development cycles and TTM (Time to Market).
- Companies need to be able to modify their own software and systems, which fully fits into the concept of open source software, which has become very successful.
- CPU clocks are barely increasing, but multi-core processors are standard, and networks are getting faster. This means parallelism is only going to increase.
- Companies often benefit from outsourcing server capacities. Amazon Web Services and Microsoft Azure offer highly demanded cloud infrastructures, such as IaaS (Infrastructure as a Service).

Creating software systems is challenging. And the transition to a parallel pattern of working with data requires special skills and attention to the software being developed. From this point of view the following systems characteristics are especially important:

- Reliability.
- Scalability.
- Maintainability.

Let us dwell on each of these aspects in more detail, give their definitions and justify their importance for the success of corporate information systems in general.

2. Characteristics of Enterprise Systems

2.1. Reliability

The concept of reliability aggregates in itself such characteristics as the ability to recover from failures, resistance to hacker attacks and the consistent level of performance with user errors, software and hardware faults.

Thus, the system is reliable if it works correctly. There is no sense in creating an application that will survive the destruction of all servers on which it is deployed. However, one must strive to handle known types of failures in the early stages of system development. It is much more difficult to comprehend and rewrite a large system than display a bit of healthy paranoia at the start of work.

In terms of hardware faults, reliability is closely related to scalability, which is discussed further. Scaling data storage, in addition to increasing bandwidth, also protects against data loss due to hard drive faults. Apart from data duplication between data processing centers (DPCs), RAID is also used on a server scale. The problem of hard disk failure seems not so important given the probability of a 1% failure during the first 3 years of operation [41]. But on a storage cluster with 10,000 disks, we should expect on average one disk to die per day.

Not all hardware faults are related to the problem of data storage failure, but loss of information can have huge costs in terms of lost revenue and damage to reputation [13].

One can find software errors just looking at the application code, but the devil is in the details. Such factors as the size of the application code, qualification and perseverance of those who check it, integration with third-party services and the problem of race conditions [42] reduce to zero the theoretical possibility to detect all errors before running the code. Often, software errors lie dormant for a long time until they are triggered by an unusual set of circumstances.

Thus, to ensure the reliability of the system being developed, at a minimum, it is necessary to test the logic of the application and the source code itself, as well as back up data and store the complete log of its changes. Both of these requirements should be fulfilled starting from an early development time frame, and embedded in the overall process on an ongoing basis.

2.2. Scalability

There are two types of scalability of information systems: vertical and horizontal or shared nothing [32].

With vertical scalability, a possible increase in throughput and performance is achieved by increasing the capacity of an individual server, for example, by increasing RAM size or replacing the processor with a more powerful one. Horizontal scalability implies that adding a new server increases the load that the system can handle. The term shared nothing well reflects that the servers do not share common resources, such as CPU time, RAM, or hard drives, that is, they are independent.

On the one hand, improving the performance of a single server, instead of using multiple relatively low-power stations, has long been considered outdated, because the frequency of a single processor core no longer increases to a great extent as it did before [47].

On the other hand, there appeared new processors for desktop computers that have 16–18 cores and 32–36 threads of execution with an average frequency of one core equal to 3 GHz, which greatly exceeds the “classic” quad-core processors with a frequency of 3.2–3.6 GHz. Thus, even classical vertical scaling follows the path of horizontal scaling of the processor, increasing the number of cores rather than the frequency of their work, limited with the current architecture [19].

Anyway, it is a technical debt to consider the option of vertical scaling right from the start, in the current realities of increasing requirements for fault tolerance, throughput, the ability to perform rolling upgrades. There is no doubt about it. Even Martin Fowler in 2002 in his book discussed the architecture of horizontally scalable systems as the most preferable [17].

Thus, it is safe to call developing monolithic applications, which initially do not imply the scaling possibility, an anti-pattern. Service architecture with independent deployment of blocks and their horizontal scalability, is becoming a common approach in the development of complex information systems. It should be noted that there is no need to build a service architecture for simple applications, where such "flexibility" will bring more problems than benefits.

2.3. Maintainability

In addition to reliability and scalability, how flexible the system architecture is, how easily it adapts to changing functional and technical requirements on the part of the business, determines whether a business can be quickly rebuilt and remain competitive where the life cycle of projects is measured in years and even decades.

When a project goes into the stage of support for 3, 5, 10 years and new functionality is added rarely, developers supporting such legacy or brownfield [2] project have a desire to rewrite it again. In some cases, it is justified. When a project uses a decade-old technology stack, it becomes harder to find people who can and would like to work with it. It happens that a project that has been living for a long time needs a new functionality or a change of the old-fashioned interface, which entails rewriting the scenarios of the application server part.

Rewriting everything from scratch is not always necessary. When it comes to projects with a monolithic architecture [24], small improvements using new technologies are complicated due to the fragility of the design and code, and the lack of modularity of the system parts. The complexly expandable system can be based on a relational database without replication, where the application logic relies on the ACID guarantees (Atomicity, Consistency, Isolation, Durability) [20]. Also the minimum number of simultaneous user sessions is ideal for functioning of such a system.

In this situation, you can try to carefully divide the monolith into separate services with a limited area of responsibility and the possibility of their independent deployment. It makes no sense to break all application use scenarios into services, but gradually, as the parts of the application are affected by new functionality, it needs to be done.

The modular system of services or micro-service architecture [33] has several advantages, such as independent deployment, loose coupling, using appropriate tools, programming languages, interaction methods and databases for the solution of different tasks [36].

However, the application should not be split into micro-services just for the sake of the Single Responsibility Principle or because the approach requires it. Excessive granularity leads to new problems in support not inherent in monolithic architecture. It becomes difficult to track connection between services, user query execution flows, interaction invariants between versions of deployed services, as well as backward compatibility of contracts.

DDD (Domain Driven Design) approach to software development and dividing the application into multiple bounded contexts makes possible a transition to a single database for a single service [43].

Various factors that influence the maintainability of an application are: the technology used, the culture of writing code, coverage by tests, and architectural decisions taken in the early stages of the project life cycle. There is no easy fix for making the system reliable, scalable and maintainable. But it is absolutely necessary to control the complexity of the project and its technical debt, to adequately assess future extension points, to conduct continuous testing, to have chances not to rewrite the code every two years.

3. Data Storage

3.1. NoSQL and SQL

Having chosen programming languages, platforms, and the overall technology stack of a project, one faces the question of choosing an appropriate data storage system. Eventually, there is a choice between relational and NoSQL databases.

The concept of a relational data model was first described in the article by Edgar Codd in 1969 [9]. Thus, research and development in the field of relational databases has been going on for 50 years. It is based on a well-developed theoretical apparatus and language for building queries to SQL data (Structured Query Language), which was first standardized at ANSI (American National Standards Institute) and ISO (International Organization for Standardization) in 1986 and 1987, respectively [6].

The term NoSQL in its current interpretation was formed in 2009 [38]. This direction of databases is focused on scalability, fault tolerance, ability to perform a large number of write operations, and the concept of Eventual Consistency (EC) [44] is crucial for them. EC implies that the system data, without being updated over time, will be consistent in all replicas, and data access services will return the same last recorded value from any replica. This guarantee is much weaker compared to ACID, but such behavior allows to achieve fast recording, scalability and fault tolerance.

One of the significant differences between relational and document-oriented databases is considered to be the presence of a strict data schema in the first case and the lack of such in the second [18]. However, this is not entirely true. Relational databases do have a mandatory schema, the so-called schema-on-write, while document-oriented databases allow you to store data in any form, including unstructured data. Though it does not imply that it is possible to work productively with such data. Anyway, there is a schema, but in the form of schema-on-read, when the client code expects to receive data in a specific format. Thus, schema-on-read is a softer limitation than schema-on-write [26]. This freedom on the one hand makes it possible to support multiple versions of data schemas, to perform hot-swapping of deployed services, while the other carries the risk of data inconsistency. A more detailed comparison of the database types in question is presented in the article [18].

However, in many areas considered in the article, and in the field of data storage and processing in particular, there are tendencies to take the best from several worlds, creating hybrid solutions. For example, the Polyglot Persistence approach [16] allows not to make a choice in favor of either a relational database or document-oriented storage, but gives an opportunity to select a variety of data storage options within a single project on grounds of expedience. This approach removes the strict framework and allows you to achieve flexibility by using tools that are best suited for the needs and tasks of the application parts. Alongside Polyglot Persistence, NoSQL techniques are now being introduced into classical relational databases, such as MS

SQL Server and PostgreSQL [46]. In particular, there are new data types that can be stored in columns, for example, JSON documents (JavaScript Object Notation) [45]. But there is a reverse trend, i.e., the creation of SQL-like data access languages in NoSQL solutions, for example, for MongoDB [5].

3.2. OLTP and OLAP

In order make a conscious choice between the types of data storage, you need to know the scenarios for its use. The number of write operations per second, the main types of read requests, the types of connections between entities, the cost of data loss, the criteria of fault tolerance – all of this crucially affects the choice between databases and the way data is organized, in particular the choice between OLTP and OLAP.

OLTP (Online Transaction Processing) is a way of organizing databases, in which the system works with small-sized transactions, but with a large stream, and at the same time the client expects a minimum response time from the system [11].

OLAP (Online Analytical Processing) is a data processing technology that consists in preparing summarized (aggregated) information based on large datasets, structured according to a multidimensional principle [10].

Based on these basic definitions, OLTP databases are used for prompt processing of user requests, while OLAP solutions are used for analyzing the system's snapshot at any point in time, such as OLAP cube [34], Data Warehouse [40], or Data Lake [25]. The use of OLTP solutions is typical of business transaction scenarios [17], where user input initiates writing to the database and executing read requests.

The need to create analytical reports scanning a large volume of data, possibly entire tables, led to the emergence of OLAP solutions that support a different interaction pattern than the OLTP solutions.

The comparison of access patterns for these two classes of solutions is given in Tab. 1 [26]. Initially, the same databases were used for both transaction scenarios and creation of analytics.

Table 1. Comparing Characteristics of Transaction Processing versus Analytical Systems

Property	Transaction processing systems (OLTP)	Analytic systems (OLAP)
Main read pattern	Small number of records per query, fetched by the key	Aggregate over the large number of records
Main write pattern	Random-access, low-latency writes from user input	Bulk import or event stream
Primarily used by	End-user/customer, via web application	Internal analyst, for decision support
What data represents	Latest state of data (current point in time)	History of events that happened over time
Dataset size	Gigabytes to terabytes	Terabytes to petabytes

In this regard, SQL proved a powerful and flexible tool. However, over time, analytics was removed into separate Data Warehouses [12]. Data Warehouse receives the data gathered from all available sources in the company, which is aggregated, cleaned, transformed into a convenient format for creating analytics and loaded into the repository without editing options, i.e., for read-only. The described data loading process is called Extract-Transform-Load (ETL), that is, divided into stages: data extraction, transformation and loading [26].

While OLTP contains mostly normalized data without duplicating of information, OLAP solutions achieve their goals by denormalizing data. Thus, the number of table join operations is reduced.

The above systems are designed to solve different problems. Combining creation of analytics and reports with execution of users' business transactions may be easy at the start of the project, but give less flexibility in the long run and greatly affect the performance of both usage scenarios with increasing data. However, there are systems on the market that combine both solutions, for example, Microsoft SQL Server and SAP Hana, giving access through a common SQL interface [15, 29].

It may seem that the degradation of system performance with an increase in the amount of data when using one solution instead of two is acceptable within reasonable limits. However, Amazon's research suggests that increasing server response time by only 100 ms reduces revenue by 1% [31], other studies indicate that a 1 second slowdown reduces customer satisfaction by 16% [3, 14].

You need to cater to the needs of people who work with your systems every day throughout their working hours, increasing efficiency, reducing the time of response and report generation, not blocking the system operation with modal windows, and so on and so forth.

Storing the stream of events that occurred in the system is a fundamentally different storing data option, compared with storing only the system's current state. This option requires a different attitude and design, but the result makes it possible to receive the system status at any time, add analytics to make it as complex and deep as can be.

3.3. Event Stream Storage

Machine learning in general and its models in particular require source data, and the more extensive and high-quality it is, the more effective the work gets. This explains the importance of the source material, the lack of which may be the problem with the customary approach of storing only the last state of existing business entities.

The transition from storing only the current state without the history of changes, which would allow to restore data as of any moment of the systems existence, to storing the stream of events affecting the data, as the primary source of application data, seems to be a very logical step in the development of technology and world view of the community of programmers, architects and business in the broad sense of the word. Keeping a complete history of changes gives data tools "food for thought".

However, it is necessary to perform not only the transition to systems with events stream storing, but also the transition from a single relational data repository, or a synchronous replication repository – to which the community has become accustomed and even addicted – to a distributed repository with replication and partitioning, or sharding. The transition to asynchronous propagation of changes and abandoning distributed transactions is also a must as a guarantee of the integrity of operations, though it comes at a price in terms of overall system performance and its throughput for reading and writing.

Such shift in the mass paradigm in application development seems just as necessary and inevitable as the transition from single-core and single-thread systems and the absence of Race

Conditions [42] to a multi-thread model that imposes certain restrictions, requires greater care and accuracy, as well as the use of tools for the synchronization of threads and execution processes.

It is worth noting that despite the fact that multi-core systems and software models for working with them have been around for a long time by the IT standards, developers are still rather slow at mastering them. This may partially be due to the inertia of the process of higher education, which does not yet cover this aspect by default, as basic computer literacy, and partially due to the fact that many companies need a quick solution, designed for only a few users with certain risk of data loss and a high time of response that is considered non-critical for internal corporate users, and such approach prevails over more costly current approaches. Therefore, employees find it hard and unnecessary to learn new things, since there is no demand for such skills in the company.

Of course, the idea of storing the history of data change and obtaining its inherited representation cannot be called a new one. A similar approach is found in relational databases which store the transaction log and WAL (write ahead log) for indexes. They allow to restore the state of a database after failures and deadlocks, as well as conduct an audit, but are limited in size and are often cleared after a certain period of time, that is, not stored forever.

However, storing the entire stream of changes and events, in particular Event Sourcing, does not involve the removal of old events to save disk space. In contrast, the events that occur are considered immutable, and this has several advantages. Among such advantages we can single out the possibility of the in-depth analysis of the history of system events, creation of analytics of any complexity having a complete history, not just the latest actual state of the system, caching events, etc.

The transition to the accumulation of huge datasets has set the task of their smart analysis, identifying patterns and predicting the behavior of systems that are directly affected by feedback from end users. The practical application of the revived direction of machine learning has become the solution to these issues, without which it seems impossible to continue effective business operations.

4. Machine Learning

4.1. Relevance

The application of machine learning has literally captured the minds of IT. Predicting product demand, personalized targeted advertising, fraud detection – these are just a few of the applications that everyone has heard of. To get the idea of the great demand for specialists in this field, it is enough to analyze data from hh.ru website. The current labor market demonstrates a high demand for skills and technologies directly related to big data, analysis and machine learning [21]. Python, Big Data, Machine Learning, Hadoop, Spark, Data Mining, Deep Learning, Scala are at the top of the list.

The hype around this new and exciting – by IT standards – field, attracts more and more young professionals. But besides yesterday's students, developers with experience are also eager to try their hands at the field of intelligent systems. Interest in another "breakthrough" framework for building server or client-side of applications fades as time goes by, some new technologies appear, and there is no bottom to this turmoil. However, when it comes to big data and machine learning, this new, previously unavailable business tool gives a bonus unattainable before – new knowledge. Therefore, the relevance of this direction will continue to increase with the development of models and methods.

The analysis of historical data allows us to understand where the business is losing money, to identify hidden trends and relationships, to avoid unprofitable decisions using existing experience.

Stream processing of new data allows you to make tactically more balanced decisions, increase profits in the short term, identify and prevent malicious activity, thereby reducing losses.

Any new field requires workers of a new type, and now this is the case with Data Science (DS) specialists. The thing is, data science specialists are not software engineers [1]. The key skills of engineers are programming and creating software systems, whereas the core competencies of a DS specialist are mathematical statistics, mathematics in general, machine learning and analysis. Most of all, these two profiles overlap in the area of big data.

Attempts to impose responsibilities of engineers on DS specialists lead to a loss of efficiency. The speed of solving problems associated with data analysis can reduce by 70–80% [1]. Therefore, companies need to divide these areas, distribute responsibilities and competencies of employees. But there must be a point of contact, and thus a machine learning engineer becomes such point. Usually software engineers with a set towards mathematics and 3–6 years of experience in software development and data flow design become specialists of this kind. They feel cramped in the framework of their routine tasks and see an opportunity to do what they have always wanted, but found difficult to start.

The toolkit of today allows a gradual transition from software engineering to data science, step by step, deepening the knowledge of underlying theory. Such training can take years, but it is possible that in 5–10 years we will see a legion of this kind of programmers.

In many ways, the explosive growth of the popularity of various machine learning models and distributed systems that handle large volumes of often unstructured data is caused by the advances in computer technology performance, increasing the number of processor cores, reducing the cost of data storage, including SSD technology, and most importantly the ever-growing use of cloud services as basics of available distributed computing.

4.2. Applicability

The problem of many companies is that there is not enough understanding of where and how to appropriately apply methods and models of machine learning to the available data on business processes. It is also a non-trivial task to find the sources of information from which to write the history of changes in the long-term storage for its further analysis, to identify trends, to ask the right questions, the answers to which will help reduce costs and increase the efficiency of the company as a whole.

Existing machine learning models, such as regression, classification, clustering, and neural networks, have proven themselves effective, but, as in many areas of science and technology, hybrid models are of considerable interest, those that are created at the intersection, absorbing all the best from several families of models and bringing something new, increasing the accuracy and speed of forecasting.

In Western and European countries, in particular in the Netherlands, there have long existed degree courses in data science. In Russia, however, the practical aspect of applying university knowledge in mathematical statistics, theory of probability, econometrics, and mathematical analysis is not yet so well emphasized. Such a gap between training and rapidly changing needs of the labor market adversely affects the pace of automation of business processes. In addition, not all IT specialists at this stage feel an urgent need for retraining, mastering skills related to machine learning, big data, distributed computing, algorithms used in cryptocurrencies.

Such rigidity from the part of developers is largely determined by the inertia of processes and repeated tasks, which programmers in IT departments of large companies have faced for the most part over the past 10 years. Growing companies, by contrast, can afford to adapt more quickly and try to introduce new models. It is worth pointing out that yesterday's students learn more quickly, pursue career path faster and sometimes, as early as at the age of 21, work in the position of Senior Developer.

Large corporations, which have established complex and highly inert business processes and are not largely involved into information technology and consulting in the field of data analysis, can experience considerable difficulties in the transition to new business models. Despite this, such complete rethink of information expertise seems vital.

Timely upgrade will allow to remain competitive in the market, actualize the technologies and knowledge which among other things can be used to reduce costs associated with support of outdated approaches and tools.

Conclusion

Whatever direction the enterprise information systems development may evolve in, – be it a variety of data storage solutions, evolution of processors, introduction of new software architecture – the major ubiquitous tendency is moving away from the vertical scalability to the horizontal one. With the growth of data volume and the complexity of its analysis, it is the scalability of computing and data warehousing that is becoming the cornerstone of further software development evolving. Supercomputing systems are becoming the foundation of the future digital economy of Russia and the whole world. There is a clear need for decentralized development and extensive use of supercomputers at universities and enterprises. A supercomputing complex at SUSU in Chelyabinsk is an example of such successful implementation and application [27, 28].

Despite the abundance of terms and cliched phrases, such as big data, machine learning, the Internet of things, asynchronous behavior, parallelism, distribution, they are united by common basic concepts, ideas, and problems, already known in the twentieth century. For example, abstractions like Acknowledgment or Race Condition Automaton have existed in circuit engineering for a long time, but modern IT continues to rediscover them again and again with the advent of new fashionable development technologies and paradigms.

From this perspective, modern "breakthrough" and "hype" trends seem to be somewhat a rethinking of the existing bundle of knowledge, showcasing it in a new sparkling edition. Hence, there is no "silver bullet", which would solve all problems in a certain area at the snap of your fingers. Indeed, the development of such a universal tool may require a significant rethinking of approaches to building information systems.

The inertia of introducing new approaches to development is explained by the complexity of the mental shift towards the event stream as the main option for storing data, though it is also well-grounded on the past. For instance, take an account book, where entries are made, but not corrected after being made, and errors in previous entries are only compensated by new lines.

Due to ideological, methodological and technical difficulties, enterprise systems in the vast majority of companies are built using approaches that have been approbated over the past decade.

We can say that there is a number of common tasks, such as document management, authentication, authorization, CRUD operations for various business entities (Create, Read, Update, Delete), building reports and analytics, etc. Their solutions have also been largely established, though there are various options both from the point of view of the technological stack and the quality of the implementation.

Despite the evolution of hardware and IT infrastructure, the mathematical interpretation of the domain processes has not changed so much compared to the changes in technosphere.

For example, today much is said about sensor networks and what can be achieved with their help to describe various objects of control. But at the same time, it does not matter how many sensors there are on an object that we want to control more efficiently, these sensors only record the dynamics of a control object's state, and nothing more. The controlling mechanisms themselves have not experienced any breakthrough. In essence, this process of reflecting the characteristics of a control object in time is nothing more than a digital shadow. And globally, decision-makers are still responsible for making decisions based on experience and intuition, rather than on adequate models that would allow to justify these decisions.

Modern academic research is out of touch with the practice of industry. Theoretical studies are important, but, unfortunately, are very far from practical needs.

The field of data science, entering upon the stage of IT specialists training, will very likely change, or rather, is already changing, the vector of information systems development, training specialists for their building and maintenance, and what is most important, the tasks, solving which business becomes more competitive.

It can be unequivocally asserted that big data, data science, and evolution of scalability and fault tolerance of information systems have predetermined the development of information technologies in the long term. Companies which do not take advantage of the full range of the offered opportunities can stay afloat, but only those that invest resources and adapt to rapidly changing realities will ensure maximum benefit and gain the upper hand.

References

1. Anderson, J.: Data Engineers vs. Data Scientist. `https://www.oreilly.com/ideas/data-engineers-vs-data-scientists` (2018), accessed: 2019-10-13

2. Belcham, D., Baley, K.: Brownfield application development in. NET. Manning Publications Co. (2010)

3. Brutlag, J.: Speed Matters for Google Web Search. `http://googleresearch.blogspot.com/2009/06/speed-matters.html` (2009), accessed: 2019-10-13

4. Buxton, B.: Big Data: the Next Google. `https://www.nature.com/news/2008/080903/full/455008a.html` (2008), accessed: 2019-10-13

5. Celesti, A., Fazio, M., Villari, M.: A study on join operations in MongoDB preserving collections data models for future internet applications. Future Internet 11(4), 83 (2019), DOI: 10.3390/fi11040083

6. Chamberlin, D.D.: Early history of SQL. IEEE Annals of the History of Computing 34(4), 78–82 (2012), DOI: 10.1109/MAHC.2012.61

7. Cielen, D., Meysman, A.D.B., Ali, M. (eds.): Introducing Data Science. Manning (2016)

8. Clarke, T.: Twenty Two Plus Instagram Stats That Marketers Can't Ignore This Year. `https://blog.hootsuite.com/instagram-statistics` (2019), accessed: 2019-10-13

9. Codd, E.F.: Derivability, redundancy and consistency of relations stored in large data banks. IBM Research Report, San Jose, California RJ599 (1969)

10. Codd, E.F., Codd, S.B., Salley, C.T.: Providing OLAP (on-line analytical processing) to user-analysts: an it mandate (1992)

11. Conn, S.S.: OLTP and OLAP data integration: a review of feasible implementation methods and architectures for real time data analysis. In: Proceedings of the IEEE SoutheastCon 2005, 8-10 April 2005, Ft. Lauderdale, FL, USA. pp. 515–520 (2005), DOI: 10.1109/SECON.2005.1423297

12. Dedi, N., Stanier, C.: An evaluation of the challenges of multilingualism in data warehouse development. In: Proceedings of the 18th International Conference on Enterprise Information Systems, 25-28 April 2016, Rome, Italy. SCITEPRESS - Science and and Technology Publications (2016), DOI: 10.5220/0005858401960206

13. Drinkwater, D.: Does a Data Breach Really Affect Your Firm's Reputation. `https://www.csoonline.com/article/3019283/does-a-data-breach-really-affect-your-firm-s-reputation.html` (2016), accessed: 2019-10-13

14. Everts, T.: The Real Cost of Slow Time vs Downtime. `http://www.webperformancetoday.com/2014/11/12/real-cost-slow-time-vs-downtime-slides`, accessed: 2019-10-13

15. Färber, F., May, N., Lehner, W., et al.: The SAP HANA database – an architecture overview. IEEE Data Eng. Bull. 35(1), 28–33 (2012), `http://sites.computer.org/debull/A12mar/hana.pdf`

16. Fowler, M., Sadalage, P.: The Future is Polyglot Persistence. `https://martinfowler.com/articles/nosql-intro-original.pdf` (2012), accessed: 2019-10-13

17. Fowler, M.: Patterns of enterprise application architecture. Addison-Wesley Longman Publishing Co., Inc. (2002)

18. Gaikwad, R.G., Goje, A.: SQL and NoSQL: Which is better. International Journal of Emerging Technologies and Innovative Research 2(8), 3277–3284 (2015), `http://www.jetir.org/papers/JETIR1508005.pdf`

19. Gepner, P., Kowalik, M.F.: Multi-core processors: New way to achieve high system performance. In: International Symposium on Parallel Computing in Electrical Engineering, 13-17 Sept. 2006, Bialystok, Poland. pp. 9–13 (2006), DOI: 10.1109/PARELEC.2006.54

20. Gray, J.: The transaction concept: Virtues and limitations. In: Proc. of the 7th Int. Conf. on Very Large Databases, 13-17 Sept. 1981, Cannes, France. pp. 144–154 (1981)

21. Grishina, A.: Big Data and Data Science Labor Market Survey. `https://habr.com/company/newprolab/blog/320336` (2017), accessed: 2019-10-13

22. Hilbert, M., López, P.: The world's technological capacity to store, communicate, and compute information. Science 332(6025), 60–65 (2011), DOI: 10.1126/science.1200970

23. Jun, S.P., Yoo, H.S., Choi, S.: Ten years of research change using Google Trends: From the perspective of big data utilizations and applications. Technological Forecasting and Social Change 130, 69–87 (2018), DOI: 10.1016/j.techfore.2017.11.009

24. Kalske, M., Mäkitalo, N., Mikkonen, T.: Challenges when moving from monolith to microservice architecture. In: Current Trends in Web Engineering - ICWE 2017 International Workshops, Liquid Multi-Device Software and EnWoT, practi-O-web, NLPIT, SoWeMine, 5-8 June 2017, Rome, Italy, Revised Selected Papers. pp. 32–47 (2017), DOI: 10.1007/978-3-319-74433-9_3

25. Khine, P.P., Wang, Z.S.: Data lake: a new ideology in big data era. In: ITM Web of Conferences. vol. 17, p. 03025. EDP Sciences (2018), DOI: 10.1051/itmconf/20181703025

26. Kleppmann, M.: Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly (2016), `http://shop.oreilly.com/product/0636920032175.do`

27. Kostenetskiy, P., Safonov, A.: SUSU supercomputer resources. In: Proc. of the 10th Annual Int. Scientific Conf. on Parallel Computing Technologies, PCT 2016, 29-31 March 2016, Arkhangelsk, Russia. CEUR Workshop Proceedings. vol. 1576, pp. 561–573 (2016)

28. Kostenetskiy, P., Semenikhina, P.: SUSU supercomputer resources for industry and fundamental science. In: 2018 Global Smart Industry Conference, GloSIC, 13-15 Nov. 2018, Chelyabinsk, Russia. pp. 1–7. IEEE (2018), DOI: 10.1109/GloSIC.2018.8570068

29. Larson, P., Clinciu, C., Fraser, C., et al.: Enhancements to SQL server column stores. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, 22-27 June 2013, New York, NY, USA. pp. 1159–1168 (2013), DOI: 10.1145/2463676.2463708

30. Li, C., Gu, J.: An integration approach of hybrid databases based on SQL in cloud computing environment. Software: Practice and Experience 49(3), 401–422 (2019), DOI: 10.1002/spe.2666

31. Linden, G.: Make data useful. `https://www.scribd.com/doc/4970486/Make-Data-Useful-by-Greg-Linden-Amazon-com` (2006), accessed: 2019-10-13

32. Liu, C.Y., Shie, M.R., Lee, Y.F., et al.: Vertical/horizontal resource scaling mechanism for federated clouds. In: 2014 International Conference on Information Science Applications, 6-9 May 2014, Seoul, South Korea

33. Namiot, D., Sneps-Sneppe, M.: On micro-services architecture. International Journal of Open Information Technologies 2(9), 24–27 (2014)

34. Naouali, S., Salem, S.B.: Towards reducing the multidimensionality of OLAP cubes using the evolutionary algorithms and factor analysis methods. CoRR abs/1602.04613 (2016), `http://arxiv.org/abs/1602.04613`

35. Novikov, D.A.: Big Data and Big Management. `https://mipt.ipu.ru/sites/default/files/page_file/BigDataBigControl.pdf`, accessed: 2019-10-13

36. Richter, J.: Architecting Distributed Cloud Applications. `https://www.youtube.com/watch?v=xJMbkZvuVOO` (2017), accessed: 2019-10-13

37. Robinson, H.: The Elephant Was a Trojan Horse: On the Death of Map-Reduce at Google. `https://www.datacenterknowledge.com/archives/2014/06/25/google-dumps-mapreduce-favor-new-hyper-scale-analytics-system` (2014), accessed: 2019-10-13

38. Sadalage, P.: NoSQL distilled: a brief guide to the emerging world of polyglot persistence. Addison-Wesley, Upper Saddle River, NJ (2013)

39. Safaei, B., Monazzah, A.M., Bafroei, M., et al.: Reliability side-effects in Internet of Things application layer protocols. In: 2017 2nd International Conference on System Reliability and Safety, 20-22 Dec. 2017, Milan, Italy. pp. 207–212. IEEE (2017), DOI: 10.1109/ICSRS.2017.8272822

40. Singh, S.: Data warehouse and its methods. Journal of Global Research in Computer Science 2(5), 113–115 (2011)

41. Su, C.J., Huang, S.F.: Real-time big data analytics for hard disk drive predictive maintenance. Computers & Electrical Engineering 71, 93–101 (2018), DOI: 10.1016/j.compeleceng.2018.07.025

42. Unger, S.H.: Hazards, critical races, and metastability. IEEE Transactions on Computers 44(6), 754–768 (1995), DOI: 10.1109/12.391185

43. Viggiato, M., Terra, R., Rocha, H., et al.: Microservices in practice: A survey study (2018)

44. Vogels, W.: Eventually consistent. Communications of the ACM 52(1), 40 (2009), DOI: 10.1145/1435417.1435432

45. Šimec, A., Maglii: Comparison of JSON and XML data formats. In: Central European Conference on Information and Intelligent Systems (2014)

46. Vyawahare, H., Karde, P., Thakare, V.M.: A hybrid database approach using graph and relational database. In: 2018 International Conference on Research in Intelligent and Computing in Engineering, 22-24 Aug. 2018, San Salvador, El Salvador. pp. 1–4 (2018), DOI: 10.1109/RICE.2018.8509057

47. Zhislina, V.: Why the frequency does not increase. `https://software.intel.com/en-us/blogs/2014/02/19/why-has-cpu-frequency-ceased-to-grow` (2014), accessed: 2019-10-13