International Journal of Technology 11(6) 1125-1135 (2020)
Received August 2020 / Revised November 2020 / Accepted November 2020

International Journal of Technology

http://ijtech.eng.ui.ac.id

Information Analysis Support for Decision-Making in Scientific and Technological Development

Boris Onykiy [1], Evgeniy Antonov [2*], Alexey Artamonov[1], Evgeny Tretyakov[2]

[1]*Department of Analysis of Competitive Systems, National Research Nuclear University MEPhI, Kashirskoe hwy, 31, Moscow, 115409, Russian Federation*

[2]*Laboratory of Advanced Storage and Processing Systems for Ultra Large Data, Plekhanov Russian University of Economics, Stremyanny lane 36, Moscow, 117997, Russian Federation*

Abstract. This paper presents the development of an information and analytical system to foster scientific and technological development in a given scientific field. In this work, the main software tools for implementing distributed computing, which involves a set of software components for collecting, processing, and analyzing large amounts of data, are considered. In addition, various approaches for task coordination between different sets of software are discussed and techniques for storing large amounts of data are described. The system architecture and database schema are designed and tested. Nowadays, the intellectualization of individual software agents is a key aspect of a new generation of multiagent systems. For this reason, this paper develops an approach that can organize activities of a large number of software agents to increase system intellectualization through swarm intelligence at the level of individual agents. Three remote servers were used to build and test the system deployment, comprising such components as a platform for monitoring and scheduling workflow, data storage, and a graphical user interface that enables data retrieval and interaction on the Internet.

Keywords: Apache airflow; Data collection; Data storing; Distributed computing; Multiagent system

1. Introduction

In the process of viable decision-making in scientific and technological development, a synoptic view is required regarding the current state of the specific areas of concern and the trends of modern development. In the course of performing search operations, an analyst has to interact with various sources of information, mostly located on the Internet (Berawi, 2018b). The conditions for a quick search in a short period of time determine the impossibility of performing the corresponding work in manual mode, as, in this case, aggregating a large number of unrelated information sources is necessary (Kulik, 2015; Inkina et al., 2019). In this regard, automation is needed in the processes of searching, collecting, and aggregating information (Berawi, 2018a).

In this paper, the automation of data collection and processing is achieved by developing a multiagent system (MAS). In general, an agent in information technology (a software agent) is a computer program that is activated on schedule or by request with

*Corresponding author's email: eantonov@kaf65.ru, Tel.: +74-99-2846460(8414)
doi: 10.14716/ijtech.v11i6.4465

some autonomy to perform specific tasks (Ananieva et al., 2015; Onykiy et al., 2017).

In contrast with the classical method of problem-solving (searching for a deterministic algorithm that allows the best solution to be found), in multiagent technologies, the solution is obtained as a result of the interaction among many independent targeted software agents. A review of domestic and foreign manuscripts shows the relevance of an automated data-based decision-making information system and software, and the intellectualization of an individual software agent is a key aspect of a new generation of MAS. For this reason, modern approaches for storing and analyzing large amounts of data are set forth here to consider a software-based solution for agent interaction in the distribution of data-collection and -processing tasks. Furthermore, this method aims to take into account the possibility of increasing each software agent's intellectualization.

2. Methods

In this study's aim to elaborate an architecture and design model for collecting, processing, and storing large amounts of data, modern software-based solutions have been considered. During the review of these solutions, the advantages and disadvantages of the selected methods for developing an MAS for information and analytical support were highlighted. An MAS is a group of interacting software agents that share a common goal and have the following properties: autonomy, sociality, reactivity, and proactivity (Artamonov et al., 2014).

2.1. Software Agent Communication

The sociality property of an MAS determines the need for a communication language—a protocol (or open specification) for solving the problem of interaction among agents within a system—which is used to communicate task distribution among many software agents. Messaging enables the assignment of tasks and getting the current work status of individual software system elements. Work status is a tool for obtaining information about the presence of an element and its readiness to perform a task. To implement these tasks, there are protocols that differ in their architecture and ways of interacting. The main requirements for these types of protocols are defined in the Internet information document RFC 2779 ("Instant Messaging/Presence Protocol Requirements"). This paper considers two main messaging protocols, XMPP and AMQP (Bezerra et al., 2018), and two software solutions that use these protocols, SPADE and RabbitMQ, respectively.

The Extensible Messaging and Presence Protocol (XMPP) is an open extensible protocol for exchanging messages and presenting information that uses the XML markup language. Interaction between clients is performed via the XMPP server and is asynchronous, and it transmits discrete information units (XML stans) inside XML streams.

The Advanced Message Queuing Protocol (AMQP) is an open standard for asynchronous secure and reliable transmission of messages between two parties. The main idea of AMQP is to communicate between independent network elements using message-broker software that routes and distributes data flows using a set of queues (Hong et al., 2018). A message broker is binding software that accepts and returns messages between individual software elements of the system.

To compare the protocols, we have solved the subproblem of interaction between two software agents in which one software agent assigns another agent the task of extracting data from a given information resource on the Internet. Based on the subproblem results, the features of using the protocols in the development of the information analysis support system are presented in Table 1. In the course of performing the subproblem of data collection, it was concluded that, to use the SPADE platform for agent communication, the

ID must be specified. Thus, we need to explicitly assign to whom exactly the message should be sent, as well as monitor a well-defined communication model between software agents. As a result, the scalability of the system becomes difficult.

Table 1 Comparison of SPADE and RabbitMQ (XMPP and AMQP specifications) features

Feature	SPADE	RabbitMQ
Determine the work status	Yes	Yes
Communication model	Message dispatching is performed using the ID of the agents and XML streams	Messages are published through the exchange element to the queues
Instant messaging	Agent uses the templates to dispatch received messages	Consumers pull messages from queues on demand
Message loss	Yes	No
Asynchrony	Yes	Yes
Web interface for monitoring processes	Yes	Yes

Based on the above analysis, the optimal solution is to choose the AMQP specification and the RabbitMQ message broker, as this uses message queues for interaction (Figure 1 and the scalability of the MAS is not time-consuming.

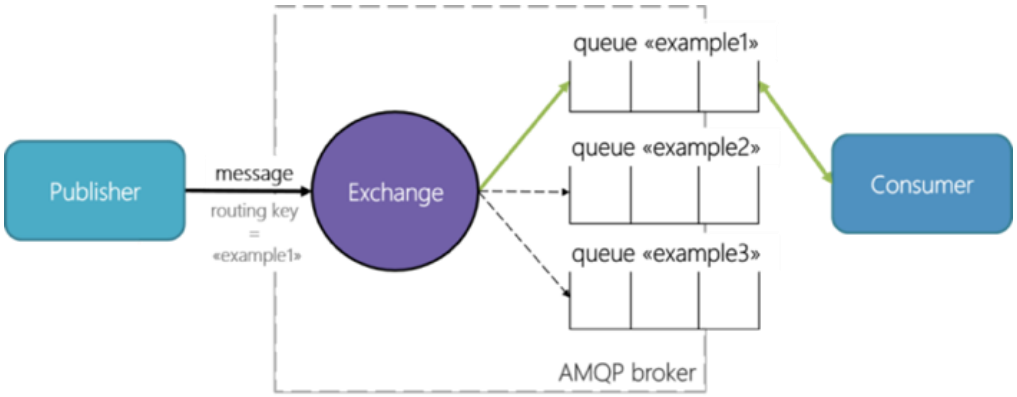

Figure 1 Model of message exchange using AMQP specifications

2.2. Workflow Planning and Monitoring

The RabbitMQ platform was selected to coordinate software agents in the information and analytical MAS. However, it was not enough to simply develop an MAS. To achieve the goal of the work, a data extraction method and a scheduler (Natesan and Chokkalingam, 2019) for distributing data collection tasks were needed, along with a tool for monitoring the execution of individual processes throughout the workflow.

2.2.1. Data extraction

The web page of an information source on the Internet is presented in the form of HTML markup. The resource content is displayed in a convenient view for the person using the web browser. When interacting with an information source, a software agent makes requests using the HTTP Protocol and the URL source to obtain and process the HTML document. The process of extracting data from HTML markup tags and attributes uses the

regular expression method and the XML Path (XPath) query language, which provides data from individual parts of the HTML markup (Yang, 2019).

Modern web services are protected from data copying. A web service can recognize the agent's behavior and block it if there are too many requests. In addition, information sources use JavaScript to generate the page's HTML code. In this case, accessing the resource content directly through a request by a URL through the program method is not possible but must be done through a web browser, where JavaScript is executed when the web page loads. One of the main tools for automating the web browser's actions and performing the above operations (i.e., managing the browser using software methods) is the Selenium WebDriver library. Using this package can imitate human behavior and simplify data collection from information sources on the Internet.

Taking into account the features of agent technologies, the Python 3.7 programming language was chosen for the development of a software agent for extracting data from informational web resources, which has a set of libraries necessary for implementing data collection (requests, selenium, lxml, and beautifulsoup4 packages) (You and Wang, 2019; Antonov et al., 2020).

2.2.2. Data collection workflow

Special attention was paid to the specific requirements of the task choosing the manager to implement workflows. The requirements include the ability to monitor the execution process in real time, the availability of a task scheduler, the scalability of the system, and the use of distributed computing. Based on this, Apache Airflow was chosen. Apache Airflow is an open-source platform for development, planning, and monitoring workflows using the Python programming language (Mitchell et al., 2019). The architecture can be represented in two types: single-node or multinode. The multinode type allows concurrent execution of tasks on several working machines, and the system can be horizontally scaled with the addition of new workers using this mode. Based on the task, the following type of architecture was chosen (Figure 2).

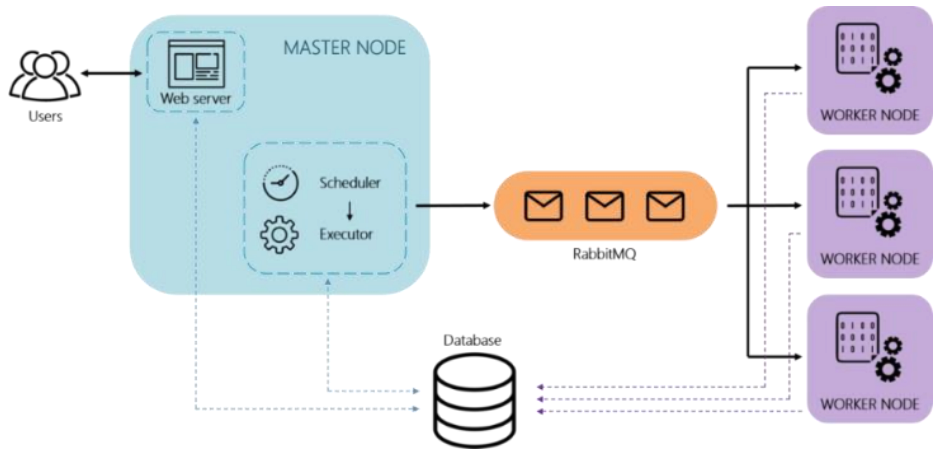

Figure 2 Multinode architecture of the Apache Airflow platform

The workflow is a directed acyclic graph (DAG) that consists of a set of tasks. A directed graph means that the tasks are performed in a certain order, while an acyclic graph means that the tasks cannot be looped. The workflow configuration data are also passed to the DAG: name, owner, execution schedule, task configuration, and so on. The DAG is defined in a Python file in a special platform folder and describes the order of execution—sequential or parallel—that is, it specifies in which order separate tasks of the process should be executed.

The Scheduler is a core of Apache Airflow and manages the launch, performance, and storage of DAGs. The Scheduler is a multithreaded process that checks and analyzes all code present in a special folder. The Scheduler also delegates workflow execution to the Executor, monitors which tasks need to be completed, and determines whether tasks need to be repeated in case of a failure. The Executor in Apache Airflow is responsible for executing tasks and defines the method for distributing tasks in the workflow.

Thus, workflow execution starts with the Scheduler, which creates a DAG instance in the database with the scheduled state (runtime and input data). Then, the Scheduler uses the management module to send workflow tasks to the queue of the RabbitMQ broker. Workers accept tasks, execute them, and transmit data about the execution status to the database. The Scheduler, based on the information in the database, decides whether to repeat a single task in case of a failure or to end the workflow in case of successful completion of all tasks.

This approach adds functionality to the workflow; for example, each aggregator can classify articles and inform the person about finding up-to-date information on the subject of interest (Fomina et al., 2020). Therefore, we can organize the activities of a large number of software agents, thus increasing system intellectualization.

2.3. Data Storage

Storage (or database [DB]) is an information model that enables storing information about a group of objects. The DB model refers to the formal representation, that is, the logical structure of the database, as well as methods for processing and organizing data. Information systems that implement automated data collection and processing depend on the representation of this data and use different types of DBs. There are two main types of databases: relational (SQL) and nonrelational (NoSQL) databases (Golosova et al., 2015).

The major problem, in this case, is the processing of a large amount of information, which requires the software tool to have a high performance and methods for searching target data using text information analysis technologies. Thus, there are two basic requirements for choosing the data storage: (1) the ability to use a dynamic data storage structure that has high performance with an increasing volume of data; and (2) the availability of methods for analyzing text information.

The NoSQL database has a significant advantage in analyzing large amounts of data, as it uses a dynamic data structure, and changing it does not require complex decisions (Grigorieva et al., 2016). Many of these types of databases are also distributed systems. However, using a NoSQL solution has a drawback, as the format of the original text is lost due to the indexing of the downloaded data. Thus, the source text of the collected documents may need to be stored separately. Based on the considered features and requirements defined for data storage, we concluded that the use of a NoSQL database for storage and analysis improves performance with an increasing amount of information in the course of data collection from information sources on the Internet. Moreover, the concept of a NoSQL solution provides the possibility of horizontal scaling of the system through distributed computing. Management systems of the NoSQL database have a set of methods for analyzing and processing data, as well as optional software for visualizing the results of the analysis. Relational data storage solutions can be considered auxiliary storage for raw data.

Elasticsearch, which is an open-source distributed software search engine that uses JSON REST API to interact with uploaded data, was chosen as the NoSQL storage (Dhulavvagol et al., 2020; Han and Zhu, 2020). Processed data in Elasticsearch is stored as an inverted index. An inverted index is a data structure in which each word corresponds to a list of document names and the position where the word occurs (Fedorova et al., 2019). A

single record structure can be represented as "word document: position". This view makes it easier to find all documents that contain a single word. When new documents are added, the index of each word is updated if necessary (Gao et al., 2012). Another feature when loading data is the use of the morphological method, which consists of shortening a word to its root; that is, during document indexing, the root word is saved instead of the actual one. The search engine also uses a synonym dictionary to search for words with the same meaning, which reduces the index size and eliminates differences between similar words. By saving the word positions in the document, not only individual words but also phrases in documents can be searched for. This is achieved by comparing positions in the uploaded documents; if the document contains words in the same order as in the query, then the phrase is appropriate (Bhatnagar et al., 2020).

Several software applications have been developed for this service to simplify working with large amounts of data. One of these is Kibana, which is an open-source plugin for visualizing and analyzing data from Elasticsearch (Shah et al., 2018). Kibana is used for searching, viewing, and interacting with data stored in the indexes of Elasticsearch.

2.4. Distributed Computing for System Implementation

The selected set of software components enables the implementation of an information and analytical support system that corresponds to the main properties of an MAS. The system architecture was designed based on the selected software tools and their features (Figure 3). The architecture was distributed and divided into three main blocks: Control center, Processing center, and Storage center. Each system block is deployed on its own remote server (at the Joint Institute for Nuclear Research and Plekhanov Russian University of Economics).

The Control and Processing centers represent the Apache Airflow platform. Workers of workflow tasks are located in both nodes. The Airflow metadata storage database and the RabbitMQ broker are deployed on the Control center. After distributing the task through the message queue of the RabbitMQ broker, the worker, if necessary, interacts with the main source of the Internet to collect data using the software packages discussed earlier.

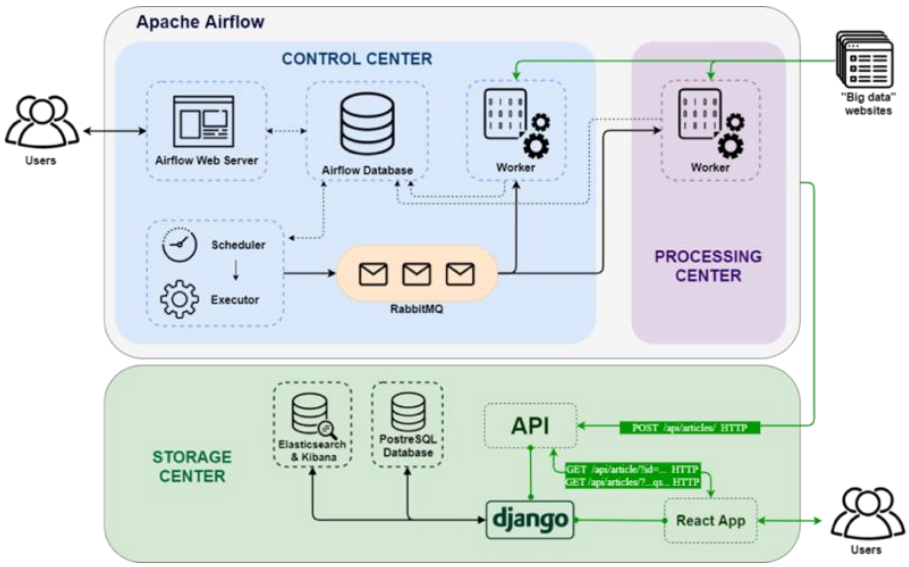

Figure 3 Architecture of a MAS for information analytical support

The Storage center contains a Django module that interacts with SQL and NoSQL data storage and provides a graphical user interface and API for working with the databases. The system's graphical user interface makes it possible to display the collected data to the user

for a requested query. Thus, this architecture and the software tools used allow the development of information and analytical MAS and obtain the necessary information on a given topic using automated data collection by a set of software agents that work together to achieve a common goal.

2.5. Algorithms for Collecting and Presenting Data

To develop a workflow on the Apache Airflow platform, a general algorithm for collecting data from an information source on the Internet was compiled (Figure 4a). The processes in gray run on Control or Processing centers (depending on how tasks will be distributed on the platform Apache Airflow), and the processes in green run on a remote server in the Storage center. The first step is to collect all the URLs of articles in the information resource, and the only URLs selected are those for which the data have not been extracted. If there are such addresses, the collection is carried out. The collected data are uploaded to SQL storage, a PostgreSQL database in the Storage center remote server, where each individual new object is assigned an ID. The identified data are uploaded to Elasticsearch. Thus, the system stores the same data in two instances: "raw" data in a SQL database and processed data in a search NoSQL store.

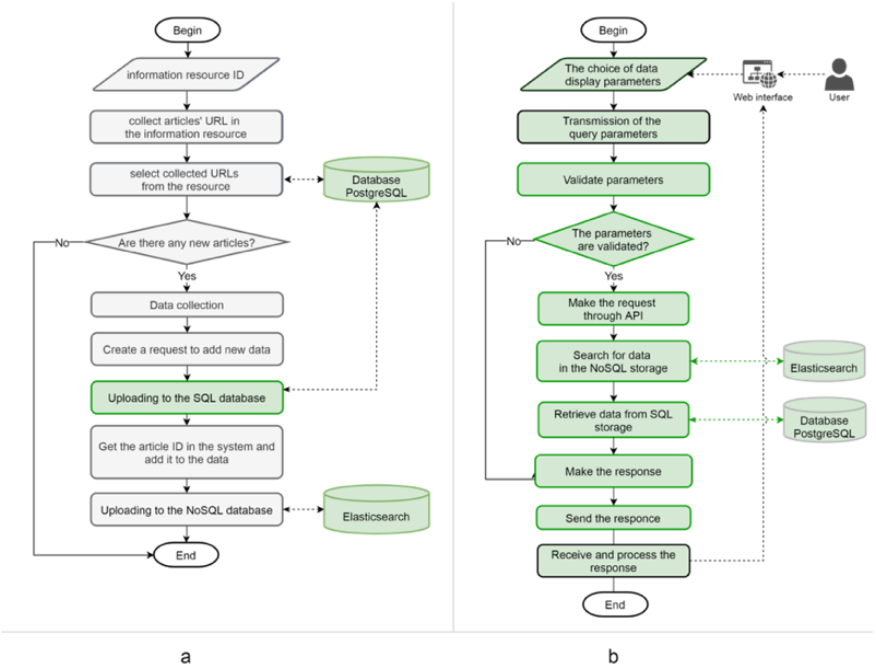

Figure 4 Algorithms for (a) collecting and (b) presenting data

The general algorithm for getting data corresponds to Figure 4b. The processes marked in green with a black outline refer to the client side of the user interface, and those with a green outline refer to the back end of the system. After the user has selected parameters for displaying data, a request is sent to get data from the storage. If the parameters pass validation, the extraction algorithm is executed; otherwise, a response with the error content is generated. To search for a dataset with the user-defined parameters that the Elasticsearch is using, the data are formatted in JSON and sent for display to the client.

3. Results and Discussion

3.1. Data Collection

The workflow is developed on the Apache Airflow platform using the Python programming language. As mentioned above, the workflow is represented as a DAG. As an

example, we took the online source MPDI (the scientific journal *Big Data and Cognitive Computing*) as an information resource. Based on the data collection algorithm shown in Figure 4a, the "newsblock_mdpi" workflow is developed to extract information from the resource. Figure 5 shows the process of completing tasks and the status of their completion at each launch.

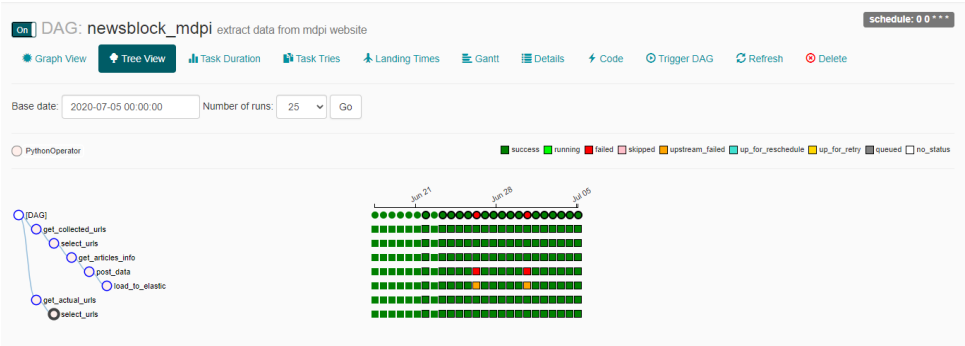

Figure 5 The completion status of the DAG "newsblock_mdpi" at each launch

The DAG consists of six tasks. The first two tasks are performed in parallel, and the rest are performed sequentially. The workflow nodes correspond to the algorithm processes. The "newsblock_mdpi" workflow is set to run on a schedule via cron (a computer program in UNIX-class systems used to periodically run tasks at certain times). The cron value is "0 0 * * *", which means the daily execution of the process at 00:00 hours.

3.2. Data Storage and Analysis

The "bigdata" index was created in Elasticsearch, and 113 objects were loaded into it by the last task of the "newsblock_mdpi" workflow. Not only was the issue of data collection solved, but the possibility of the intellectual development of the system was also provided, as software agents use one storage (i.e., shared knowledge or swarm intelligence).

Kibana includes a Dashboard that enables information about the general state of the scientific field in time to be obtained using the search bar. The result is shown in Figure 6.

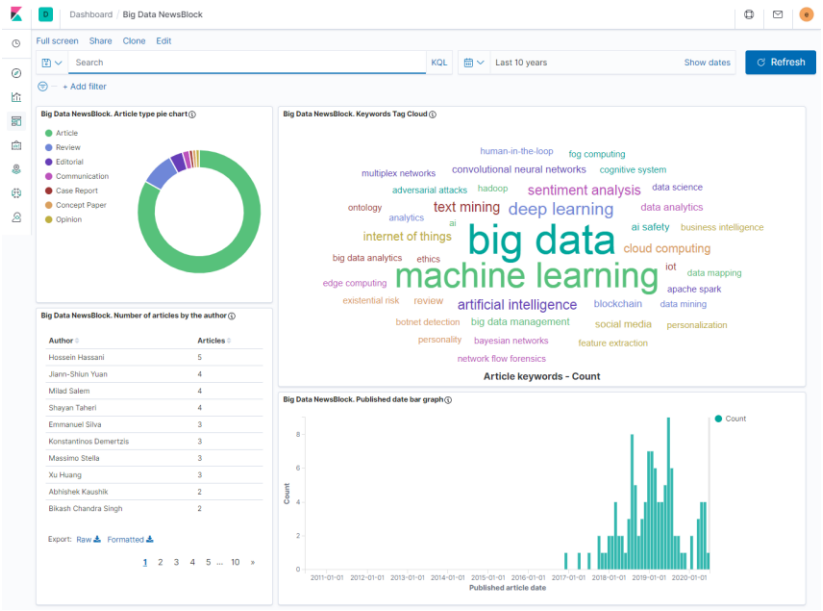

Figure 6 Dashboard for the "bigdata" index

Figure 6 displays four blocks of summary information: a pie chart representing the percentage of article types; a table with information on the number of articles written by each author; a tag cloud of the articles' keywords; and a histogram showing the statistic of articles published by day. That information allows the person to understand the state of the field and to identify the main topics that are currently developing in the field, the most active authors, where the information is most often found, and how relevant the topic is at present. The graphical interface of the information and analytical MAS is developed using the React framework, which is an open-source JavaScript library for developing user interfaces. The result of developing the graphical user interface (GUI) is shown in Figure 7.

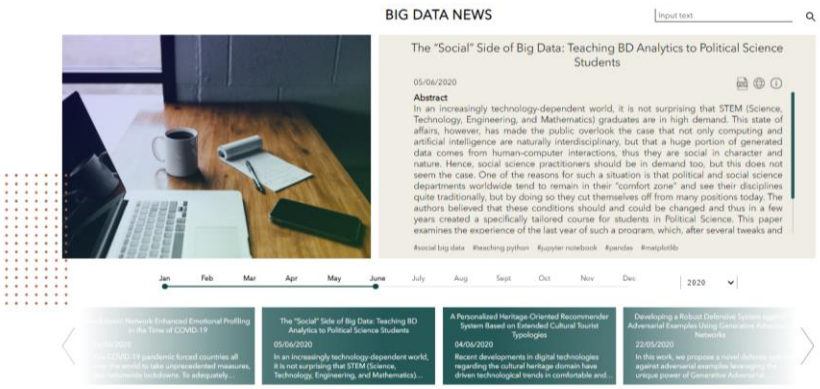

Figure 7 GUI for information and analytical MAS

The graphical interface has functionality for displaying a set of articles by a specified filter at the bottom of the page, and when one of the articles is selected, it is displayed in the main block (beige area) for viewing. A bar with a choice of months and publication date are dynamic and available to the user for change, after changing the values of this bar, the set of articles at the bottom of the page changes.

4. Conclusions

This paper considers the approach of processing data from various Internet resources in a specific field by considering the increasing volume of data in time. At this stage, the approach can be used as a data collection and analytical tool to provide superior information on a given subject (for example, for understanding customer experiences and how consumer behavior has changed over time). Furthermore, using the approach, we can organize the activities of a large number of software agents, thus increasing system intellectualization with swarm intelligence at the level of individual agents.

The developed system for information and analytical support uses distributed capacities to solve the task. The system is scalable, to improve performance and reduce workflow execution time, and it is possible to set workers using additional capacity. Elasticsearch can be scaled horizontally as well. The architecture and the database schema have been designed and tested. To collect the data from other information sources on the Internet, the DAG is added with the parameters for extracting data. Up-to-date data will be displayed and taken into account in the Dashboard and in the GUI.

In addition, at this stage of development of the MAS, we have collected the keywords that represent the Big Data technology field. The next step of the work is to improve the approach with an automated search of relevant articles on the Internet. The collected

data will be used for prediction analysis to make a list of contemporary references in a specific area for understanding the current state of technological development.

Acknowledgements

The study was carried out at the expense of the Russian Science Foundation grant (project No. 19-71-30008, 2019).

References

Ananieva, A.G., Artamonov, A.A., Galin, I.U., Tretyakov, E.S., Kshnyakov, D.O., 2015. Algorithmization of Search Operations in Multiagent Information-Analytical Systems. *Journal of Theoretical and Applied Information Technology*, Volume 81(1), pp. 11–17

Antonov, E., Lopatina, E., Ionkina, K., Evgeniy, T., 2020. Agent Data Merging. *Procedia Computer Science*, Volume 169, pp. 473–478

Artamonov, A.A., Leonov, D.V., Nikolaev, V.S., Onykiy, B.N., Pronicheva, L.V., Sokolina, K.A., Ushmarov, I.A., 2014. Visualization of Semantic Relations in Multi-Agent Systems. *Scientific Visualization*, Volume 6(3), pp. 68–76

Berawi, M.A., 2018a. Improving Business Processes through Advanced Technology Development. *International Journal of Technology*. Volume 9(4), pp. 641–644

Berawi, M.A., 2018b. Utilizing Big Data in Industry 4.0: Managing Competitive Advantages and Business Ethics. *International Journal of Technology*. Volume 9(3), pp. 430–433

Bezerra, D., Aschoff, R.R., Szabo, G., Sadok, D., 2018. An IoT Protocol Evaluation in a Smart Factory Environment. *In:* 15th Latin American Robotics Symposium (LARS), 6th Brazilian Robotics Symposium (SBR) and 9th Workshop on Robotics in Education (WRE), pp. 124–128

Bhatnagar, D., SubaLakshmi, R.J., Vanmathi, C., 2020. Twitter Sentiment Analysis using Elasticsearch, LOGSTASH and KIBANA. *In:* 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), pp. 1–5

Dhulavvagol, P.M., Bhajantri, V.H., Totad, S.G., 2020. Performance Analysis of Distributed Processing System using Shard Selection Techniques on Elasticsearch. *Procedia Computer Science*, Volume 167, pp. 1626–1635

Fedorova, V.A., Efremov, E.A., Kolyagina I.A., 2019. Search and Index Data using Elasticsearch. *Issues of Radio Electronics*. Volume 3, pp. 74–77

Fomina, J., Safikanov, D., Artamonov, A., Tretyakov, E, 2020. Parametric and Semantic Analytical Search Indexes in Hieroglyphic Languages. *Procedia Computer Science*, Volume 169, pp. 507–512

Gao, R., Li, D., Li, W., Dong, Y., 2012. Application of Full Text Search Engine Based on Lucene. *Advances in Internet of Things*, Volume 2(4), pp. 106–109

Golosova M.V., Grigorieva, M.A., Klimentov, A.A, Ryabinkin, E.A., Dimitrov, G., Potekhin, M., 2015. Studies of Big Data Metadata Segmentation between Relational and Non-Relational Databases. *Journal of Physics: Conference Series*, Volume 664(4), pp. 1–9

Grigorieva, M.A., Aulov, V.A., Golosova, M.V., Gubin, M.Y., Klimentov, A.A., 2016. Data Knowledge Base Prototype for Modern Scientific Collaborations. *Ceur Workshop Proceedings*, Volume 1787, pp. 26–33

Han, L., Zhu, L., 2020. Design and Implementation of Elasticsearch for Media Data. *In:* International Conference on Computer Engineering and Application (ICCEA) 2020, pp. 137–140.

Hong, X.J., Sik Yang, H., Kim, Y.H., 2018. Performance Analysis of RESTful API and RabbitMQ for Microservice Web Application. *In*: 9th International Conference on Information and Communication Technology Convergence (ICTC) 2018, pp. 257–259

Inkina, V.A., Antonov, E.V., Artamonov, A.A., Ionkina, K.V., Tretyakov E.S., Cherkasskiy A.I., 2019. Multiagent Information Technologies in System Analysis. *In:* Proceedings of the 27th International Symposium Nuclear Electronics and Computing (NEC) 2019, pp. 195–199

Kulik, S.D., 2015. Model for Evaluating the Effectiveness of Search Operations. *Journal of ICT Research and Applications*, Volume 9(2), pp. 177–196

Mitchell, R., Pottier, L., Jacobs, S., Silva, R.F.D., Rynge, M., Vahi, K., Deelman, E., 2019. Exploration of Workflow Management Systems Emerging Features from Users Perspectives. *In: IEEE International Conference on Big Data 2019*, pp. 4537–4544

Natesan, G., Chokkalingam, A., 2019. Optimal Task Scheduling in the Cloud Environment Using a Mean Grey Wolf Optimization Algorithm. *International Journal of Technology*, Volume 10(1), pp. 126–136

Onykiy, B.N., Artamonov, A.A., Tretyakov, E.S., Ionkina, K.V., 2017. Visualization of Large Samples of Unstructured Information on the Basis of Specialized Thesauruses. *Scientific Visualization*, Volume 9(5), pp. 54–58

Shah, N., Willick, D., Mago V., 2018. A Framework for Social Media Data Analytics using Elasticsearch and Kibana. *Wireless Networks*, Volume 1, pp. 1–9

Yang, H., 2019. Design and Implementation of Data Acquisition System based on Scrapy Technology. *In:* 2nd International Conference on Safety Produce Informatization (IICSPI) 2019, pp. 417–420

You, X., Wang, Y., 2019. Automatic Network Application System Based on Selenium. *In*: 2nd International Conference on Computer and Communication Engineering Technology (CCET) 2019, pp. 149–153