УДК 004.23:004.272.44
DOI: 10.31673/2412-9070.2023.012126

V. O. KUZMINYKH, PhD, assoc. professor;
S. I. OTROKH, D.S. assoc. professor;
B. XU, postgraduate;
R. A. TARANENKO, senior engineer;
L. I. KUBLII, PhD, assoc. professor,
National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv

EVENT-ORIENTED ARCHITECTURE IN THE SYSTEM FOR PROCESSING LARGE DATA STREAMS

The article discusses approaches to the implementation of the microservice architecture of the system for processing large data flows when collecting information on the main event-oriented approach in the organization of managing the sequence of use of individual microservices, which provide access to information sources and the extraction of relevant data corresponding to the user's request. This is important when processing large data flows from heterogeneous information sources, especially when the task is to minimize the total time of collection and processing of large data flows. This, in turn, poses the task of minimizing the number of requests to information sources to obtain a sufficient number of units relevant to the data search request. The creation of effective systems for processing big data requires constant development of approaches to the architecture of building software applications. An approach to the construction of the software system architecture is proposed, which makes it possible to manage the selection of microservices adaptively, in accordance with the events that occur during information collection, and thus form the choice of information sources based on the evaluation of the effectiveness of obtaining relevant information from them. The event-oriented microservice architecture makes it possible to adapt the operation of the software system to the loads on individual microservices and to increase the efficiency of their work by correcting calls to information sources based on the analysis of relevant events that occur during the collection and initial processing of the received data. Based on the analysis of the current state of information collection, new instances of containers can be created to extract information from the most effective sources and, thus, improve the efficiency of the system as a whole. The use of event-oriented microservice architecture can be especially effective in the development of various information and analytical systems that have the need, according to user requests, to refer to various sources of information based on their relevance, and to process large data streams when collecting information.

Keywords: microservices; adaptation; event-driven architecture.

Introduction

With the rapid development of the information society, the task of finding ways to increase the efficiency of analytical activities in corporate information and analytical systems is becoming extremely urgent. Analysts have to operate information resources that are unprecedented in their size, complexity, dynamism and growth rates, and this forces information systems developers to look for ways to improve data processing processes and technologies.

The goal of analytical activity is to provide information needs of the organization and support decision-making. Analytical activity in various fields involves working with information, its in-depth understanding, decision-making regarding analysis in one or another situation, thematic processing of information and its visualization in analytical reports, their verification, obtaining management decisions based on new knowledge].

The availability of modern means of data storage, extraction and visualization by itself does not solve the main tasks of the corporate information and analytical system. This is impossible without a powerful analytical component, which allows you to extract from the gigantic amount of data coming from various sources, the information necessary and sufficient to successfully respond to incoming calls.

The description of the analytical component must be effective. This means that the costs of conducting it should be the lowest with optimal depth of analysis and its complexity. Therefore, the use of modern methods and tools to increase the efficiency of analytical activities is extremely important. For the design of analytical activities, it is very important to use business process analysis methods, software modeling methods, network planning and management methods, which must be implemented using the latest highly effective types of software architecture [1; 2].

The development of the architecture of information and analytical systems is characterized by the need to find and implement solutions that ensure the integration of many functions and services. Classical monolithic solutions very often do not meet the requirements of the necessary flexibility of the solution, speed of obtaining results, efficiency and their quality. Most of the classic architectural solutions were proposed already decades earlier, but the possibilities of development and implementation of new architectural solutions appeared with the development of technical capabilities and their distribution, which made them easily accessible to developers.

Analysis of existing solutions

Today, the most famous and widespread architectural solutions for the development of software tools for information and analytical systems are:

• layered architecture (Layered Architecture, LA);

• multi-level architecture (Tiered Architecture, TA);

• service-oriented architecture (Service Oriented Architecture, SOA);

• micro-service architecture (Microservice Architecture, MA);

• event-driven architecture (Event-driven architecture, EDA).

Each of them has its own advantages and disadvantages, which are due to the peculiarities of their implementation. There has been a change in the binding of solution construction from an orientation on taking into account technical capabilities to an orientation on the requirements of the solution, especially in micro-service architecture. Today, there is a significant trend towards the development of software tools that will focus on building intelligent systems of multifunctional systems.

Multi-layered architecture works on the principle of division of responsibility. It has three layers: presentation layer (Presentation Layer), business logic layer (Business Logic Layer), data transfer layer (Data Link Layer) [1].

The advantages of a layered architecture is that it is a fairly simple architecture, thanks to the division of responsibilities. At the same time, it allows to implement the protection of layers from each other and to increase controllability due to the reduction of connectivity.

Disadvantages of multilayer architecture include the fact that it does not allow for serious scaling, the monolithic structure of software tools and the mandatory passage of data through each layer.

A tiered architecture divides responsibility between the data provider and the consumer. The division of functions is implemented in a complex of software tools at different levels according to the «client-server» principle. The Request Response [2] template is used to communicate with the layers. The architecture offers both horizontal and vertical scaling. Architecture can have one, two, three or more level solutions. Starting with three-level solutions, the architecture is combined with service-oriented, which allows you to create complex models.

Service-oriented architecture uses several models that are different in structure and purpose, connecting components and programs with well-defined services. The architecture has the following main components:

• services (Services);

• service bus (Service Bus);

• service repository (Service Repository catalog of services);

• SOA security (SOA Security);

• SOA governance (SOA Governance).

Service-oriented architecture considers three types of main participants — service provider, service consumer, and service registry [3]. The basis of the service-oriented architecture is the enterprise service bus (ESB), which processes requests according to a standard protocol and data format.

Service architecture distinguishes two main types of services: ***atomic services*** (*Atomic services*) — those that are not subject to decomposition, and ***composite services*** (*Composite services*) — those that combine atomic services to build complex functionality. The SOA architecture is built on several models [2]:

• message-oriented model (Message Oriented Model, MOM);

• service-oriented model (Service Oriented Model, SOM);

• resource-oriented model (Resource Oriented Model, ROM);

• policy model (Policy Model, PM);

• management model (Management Model, MM).

Micro-service architecture is a type and development of service-oriented architecture, which is built on separate components — micro-services.

The micro-service architecture consists of the following components:

• services (Services);

• service bus (Service Bus);

• external configuration (External configuration);

• API gateway (API Gateway);

• containers (Containers).

Among the main properties of micro-service architecture can be identified: division into service components, orientation to business needs, orientation to the product rather than projects, decentralized management and data management, infrastructure automation, protection against failures. The breakdown of microservices is based on the approach of subject-oriented design DDD (Domain-driven design).

Containers in the micro-service architecture are created on the basis of templates, which represent an image formed by different levels. To coordinate transactions in the micro-service architecture, two approaches are used: choreography and orchestration. Choreography is a decentralized coordination where a microservice chooses whether or not to act based on events or messages from another microservice. Orchestration is centralized coordination when an action to be performed by a microservice is communicated to it by a separate component called an orchestrator.

The advantages of micro-service architecture are weak coupling and high isolation, modularity, low failure rate, high flexibility and scalability, easy modification of containers, acceleration of iterations, improved error handling, and more efficient data processing than in multilayer architecture.

Disadvantages of micro-service architecture are increased risks of exchange failures between services, increased complexity of coordination and manageability of services, the impact of network latency problems and other problems of distributed architecture, the need for comprehensive testing of the architecture's performance, greater time spent on implementation. Orientation to business needs has resulted in the practice of actively applying the construction of software tools in accordance with microservice architecture, especially in information and analytical systems and systems for searching and processing big data.

Event-oriented architecture is a type of software architecture that takes into account the events that occur and are analyzed during the operation of the software system, and the reaction to them. An event is interpreted as an action that initiates some message or a change in the application, or an event is a significant change in the state of the software system.

In practice, event-oriented architecture is also considered as a logical development of an adaptive superstructure over micro-service architecture. In event-oriented architecture, the focus shifts to events and how they are handled by the system architecture. The logic of the event-oriented architecture is built according to two types of topology — Broker and Mediator, named by programs as mediators that connect the generator and consumer of events. The main elements of the topology:

The main advantages of event-oriented architecture are obtaining results in real-time; shorter delays in data storage and transmission; greater bandwidth; simple scalability; high resistance to failures.

The main disadvantages of the event-oriented architecture are rather complex development and initial configuration of the software system.

Choice of implementation

Modern event-oriented micro-service architectural solutions in terms of efficiency still do not have universal solutions and in complex tasks require the involvement of a lot of additional, related knowledge, which is especially relevant in the construction of information and analytical systems. The development of this direction has a significant perspective.

New architectural solutions based on event-oriented micro-service software architecture have brought the analytical, management and control capabilities of organizations with a complex distributed structure to a qualitatively new level, giving the opportunity to build a system of search, analysis and information environment that is maximally adapted to their structure and processes. The task of developing ideologies and approaches for the most effective use of micro-service architecture in solving complex information and analytical problems remains an urgent problem, which reflects the constant increase in the amount of information that must be processed to solve user requests [4].

Event-driven software architecture is a popular approach to creating a distributed asynchronous architecture used to build complex, scalable applications. This approach is flexibly adaptable to changes in the conditions of use of the software system and can be used both for small programs and for large, complex software systems. An event-driven architecture consists of highly decoupled, dedicated event-handling components that asynchronously receive information about occurring events and process that information according to current and previous events.

Event-driven microservice architectures are the best practices for implementing today's scalable cloud applications. This architecture is ideal for creating a platform for collecting and processing large data streams.

Modern event-oriented micro-service architectural solutions consist of two main types of topology: mediator and broker. A mediator topology is typically used when you need to organize multiple steps within an event through a central mediator, while a broker mediator topology is used when you want to combine events without using a central mediator. Because the architecture characteristics and implementation strategies differ between these two topologies, it is important to understand each one in order to know which is best suited for a particular situation, reflecting the direction of development of a particular information and analytics system.

A mediator topology can be used for events with multiple steps. The topology of the mediator consists of four main elements:

- queues of events,
- mediator of events,
- event channels,
- event processors.

An event flow begins with a client sending an event to a queue that transports it to the event mediator.

The mediator then sends events asynchronously to event channels to complete each step of the process. Event processors analyze the state of event channels and perform certain business logic that corresponds to the necessary event processing [5].

The broker topology is characterized by the fact that this topology does not have a central element as an event mediator to organize the initial event. There are two types of components: broker and event processor. Each event processor is responsible for processing the event and publishing a new event to notify others of the action taken. The broker implements such elements as message queues, message topics, or a combination thereof. Topology is useful when there is a relatively simple flow of processing homogeneous events [6].

The analysis of the possibilities of these two approaches to the organization of actions in the tasks of collecting and processing big data allows us to talk about the clear advantages of the second as more flexible and such that it gives the opportunity to build adaptive architectural solutions in the implementation of software systems. In this way, it is possible to build software analytical systems that have a flexible, event-adaptive scenario for collecting and processing big data.

The topology of the event-oriented micro-service broker in the information and analytical system makes it possible to:

• create events that cause requests to individual sources of information;

• form events in accordance with the primary scenario of information collection, which determines the sequence of selection of information sources;

• adjust the information collection scenario based on the analysis of the amount of relevant information received;

• form new events that determine the further choice of information collection sources;

• save changes in information collection scenarios for further use if necessary.

An event in this case means a structured message of a fixed structure, which will be sent to a certain section in the message broker, with addresses for all services that are subscribed to update the message queue for a certain section.

At the same time, such an architecture structure makes it possible to obtain results in full in accordance with the tasks set by the analyst. No matter how complex and large the analyst's request is, and therefore the overall load on the system, in the end the analyst (user) will receive the correct data that was specified in the request.

The main advantages of the presented adaptive event-oriented architecture in the system for processing large data flows are as follows [7]:

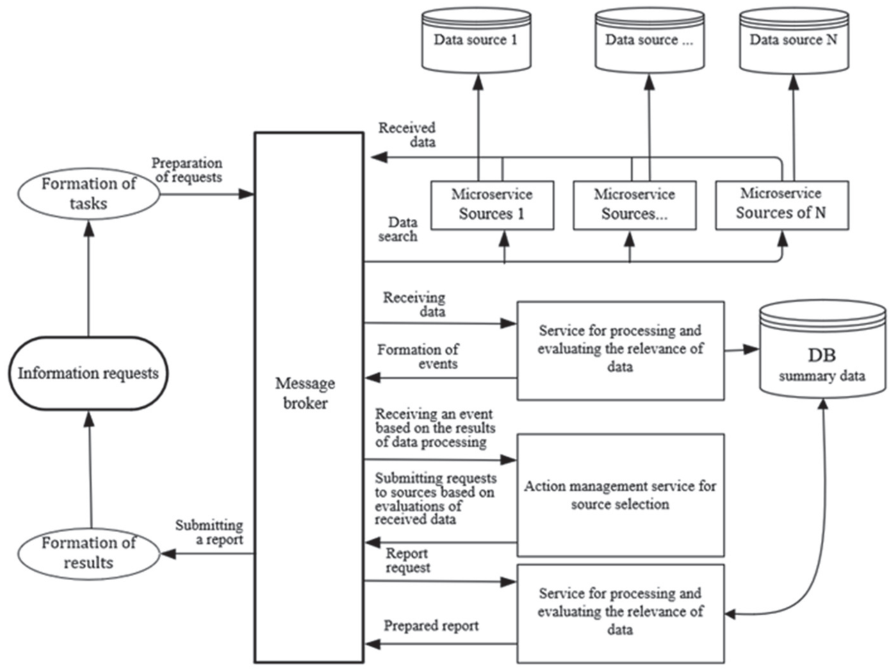
• the possibility of collecting and processing big data in real time;

• the possibility of managing parallel data processing;

• independence of system program operation from the scale of the system;

• speed of response to changes in the analyst's requests;

• the possibility of adapting the data collection scenario due to the selection of sources in accordance with the quantitative assessment of the relevance of the information obtained during the execution of the analyst's requests.


Data source 1
Data source ...
Data source N
Received data
Preparation of requests
Formation of tasks
Microservice Sources 1
Microservice Sources...
Microservice Sources of N
Data search
Receiving data
Service for processing and evaluating the relevance of data
DB summary data
Formation of events
Information requests
Message broker
Receiving an event based on the results of data processing
Submitting requests to sources based on evaluations of received data
Action management service for source selection
Submitting a report
Formation of results
Report request
Service for processing and evaluating the relevance of data
Prepared report

General architecture of the system for processing large data flows from heterogeneous sources

The development of systems for the collection and processing of large data flows from heterogeneous sources in most cases requires the determination of the specifics of specific implementations in accordance with the tasks that reflect both the specifics of the selected sources for data extraction and the specifics of the requirements for the quality, relevance and reliability of the information collected from the relevant sources.

The presented event-oriented micro-service architecture of the software system (figure on p. 25) has some features that are related to the use of a separate service that manages the actions of the system as a whole from the point of view of analyzing the performance of tasks in accordance with the primary request of the system user, as well as the selection process information sources most relevant to the request by activating the appropriate microservices associated with specific sources.

Based on the request of the system user (analytics), corresponding requests are formed, which the message broker receives. This triggers the activation process of the microservices responsible for data collection, according to the initial information collection scenario [8]. Next, a session is held to collect information from specified sources. At the end of the information collection session through the message broker, the extracted data is received by the service for processing and evaluating the relevance of the data. This service solves two main tasks:

• formation of summary data for the report;

• assessment of the relevance of the received data according to each individual source that was used in the current data collection session;

• saving summarized data in the database for further processing by the data analysis service and generating reports.

The service for processing and evaluating the relevance of data forms, on the basis of the received data, a corresponding event, which is transmitted to the service for managing actions and selecting sources.

This event is handled by the action management and source selection service and creates the appropriate requests that define the management actions for the next initiation of the corresponding microservices, which adjust the data collection scenario in the next data collection session and determine the data collection sources in the next step. The formation of such actions is based on the use of a linear stochastic automaton, which is programmed at each session in accordance with the assessment of the level of relevance of the results obtained from sources in previous sessions of data collection [9; 10].

Next, the process of data collection by the microservice from the appropriate source, which is defined by the action management service and the selection of sources in the form of requests to sources based on the received data, is repeated.

If the action management and source selection service determines that the actions on the request to receive information from the user of the large data flow processing system must be completed, then an event is generated as a request for a report to the data analysis and reporting service. This event means the end of data collection in the system, which is determined by the characteristic indicators of the user's request for information.

The report is generated in accordance with the type of request, which is determined by the corresponding template at the stage of creating tasks. The data analysis and reporting service forms, in accordance with the user's request, a report based on the data stored during information collection in the database, which summarizes the data of all information collection sessions for each user's request for information. The preparation of reports involves the secondary processing of data in order to provide them in a form convenient for use by the user.

Conclusions

The developed event-oriented architecture of the system for processing large data flows makes it possible to collect information from various forms of storage and composition of sources depending on the tasks defined by the user. The composition of information sources in this system can be expanded without interfering with other system components. Each individual microservice configured for one specific source or several homogeneous sources of information can provide only partial data, which is supplemented by other sources.

In the course of its work, the system in real time adjusts to the user's request for information in such a way that the data is taken mainly from those sources that meet the requirements of the request and can satisfy them. At the same time, source selection procedures based on a linear stochastic automaton can be used, which generates requests to sources that contain the maximum amount of data relevant to the request. The described architecture of the software system makes it possible to manage parallel data processing with significant independence of the system programs from the scale of the system.

The event-oriented architecture of the system for processing large data flows is also characterized by adaptability to the load and high throughput, which is achieved due to the use of modern approaches in the organization of the architecture of software systems.

An important advantage of the event-oriented architecture in the system for processing large data flows is also the possibility of adapting the data collection scenario due to the selection of sources in accordance with the quantitative assessment of the relay value of information received during the execution of user requests.

References

1. ***Wolff E.*** *Microservices, Flexible Software Architecture. Boston: Addison-Wesley, 2016. 436 p.*
2. ***Ashley D.*** *Bootstrapping Microservices with Docker, Kubernetes, and Terraform: A project-based guide. Shalter: Manning Publications Co., 2021. 442 p.*
3. ***Belnar A.*** *Building Event-Driven Microservices: Leveraging Organizational Data at Scale. USA: O'Reilly Media, 2020. 324 p.*
4. ***Improving*** *the Efficiency of Typical Scenarios of Analytical Activities / O. V. Koval, V. O. Kuzminykh, I. I. Husyeva [et al.] // CEUR Workshop Proceedings. 2021. Vol. 3241. P. 123–132.*
5. ***Surianarayanan C., Gopinath G., Pethury R.*** *Essentials of Mikroservices Architecture. Paradigms, Applications, and Techniques. Boca Raton: CRC Press, 2019. 314 p.*
6. ***Kuzminykh V., Koval O., Voronko M.*** *Standard Analytic Activity Scenarios Optimization based on Subject Area Analysis. CEUR Workshop Proceedings. 2019. Vol. 2577. P. 37–46.*
7. ***Rajput D.*** *Hands-On Microservices – Monitoring and Testing. Mumbai: Packt Publishing, 2018. 160 p.*
8. ***Hugo Filipe Oliveira Rocha.*** *Practical Event-Driven Microservices Architecture: Building Sustainable and Highly Scalable Event-Driven Microservices. Ermesinde: Apress, 2021. 472 p.*
9. ***Rastrigin L. A., Ripa K. K.*** *Automated random search theory. Riga: Zinatne, 1973. 344 p.*
10. ***Data*** *collection for analytical activities using adaptive micro-service architecture / V. O. Kuzminykh, O. V. Koval, S. Y. Svistunov [et al.] // Реєстрація, зберігання і обробка даних. 2021. Т. 23. № 1. С. 67–79.*

В. О. Кузьміних, С. І. Отрох, В. Ху, Р. А. Тараненко, Л. І. Кублій

ПОДІЙНО-ОРІЄНТОВАНА АРХІТЕКТУРА У СИСТЕМІ ОБРОБЛЕННЯ ВЕЛИКИХ ПОТОКІВ ДАНИХ

У статті розглянуто підходи щодо реалізації мікросервісної архітектури системи оброблення великих потоків даних під час збору інформації на основі подійно-орієнтованого підходу. Це особливо важливо в процесі оброблення великих потоків даних із різнорідних за своєю продуктивністю джерел інформації, особливо, коли стоїть задача мінімізації загального часу оброблення великих потоків даних. Пропонується підхід, який дає можливість адаптивно, відповідно до подій, що виникають під час збору інформації, керувати вибором мікросервісів, у такий спосіб формуючи вибір джерел інформації на основі оцінювання ефективності отримання релевантної інформації з них. Подійно-орієнтована система мікросервісної архітектури дає змогу адаптувати роботу системи до навантажень на окремі мікросервіси та ефективність їхньої роботи завдяки аналізу відповідних подій. Використання подійно-орієнтованої мікросервісної архітектури може бути особливо ефективним у процесі розроблення різноманітних інформаційно-аналітичних систем.

Ключові слова: мікросервіси; адаптація; подійно-орієнтована архітектура.