Petrenko V. Yu., , graduate student,
Vakaliuk T.A., doctor of pedagogical science, professor,
professor in the department of software engineering
Zhytomyr Polytechnic State University

ADVANTAGES OF MESSAGE-BASED MIDDLEWARE

In modern cloud architecture, programs are divided into small independent elements that are easier to design, deploy, and maintain. This raises the question of how best to organize the interaction of these independent modules. Message-oriented middleware are solving this problem. It is a software or hardware infrastructure that supports sending and receiving messages between distributed systems. MOM allows you to distribute application modules on heterogeneous platforms and reduces the complexity of developing applications that cover multiple operating systems and network protocols. Middleware software creates a distributed communication layer that isolates the program developer from the details of different operating systems and network interfaces using the message queue. [1] This allows software components that have been developed independently and run on different platforms to interact with each other. Programs distributed on different nodes of the network use the program interface for communication. In addition, the administrative interface makes this new virtual system of interconnected applications reliable and secure. [2]

Analogs of message-oriented middleware are remote procedure call (RPC) based middleware and object request broker (ORB) based middleware.

Messages are stored in the queue until they are processed and deleted. Message queues can be used to separate complex processing, buffering, or organize batch processing, as well as to smooth peak loads. Message queues can greatly simplify program code with disconnected components, as well as increase their performance, reliability, and scalability.

Another advantage of messaging between clients is that by adding an administrative interface, you can control and tweak performance. In this way, client programs effectively get rid of all problems, except the problem of sending, receiving and processing messages. Solving issues such as compatibility, reliability, security, scalability, and performance depends on the code that implements the MOM system and the administrator.

Many implementations of message-oriented middleware depend on the message queuing system. Some implementations allow you to provide routing logic at the very level of messaging, while others depend on client programs that provide routing information or combine both paradigms. Some implementations use broadcast or multicast paradigms.

In the middleware system, the message received at the destination does not have to be identical to the message sent first. The system can transform the message and route according to the requirements of the sender or recipient. [3] Combined with routing and broadcast / multicast capabilities, one program may send a message in its own native format, and two or more other programs may receive a copy of the message in its own format. Many modern MOM systems provide sophisticated message conversion (or display) tools that allow programmers to specify conversion rules.

The main disadvantage of many message-oriented middleware systems is that they require an additional component in the architecture, the messaging agent (message broker). As with any system, adding another component can reduce performance and reliability, and can make the system as a whole more complex and expensive to maintain. In addition, most communications between programs are synchronous, and the sender specifically wants to wait for a response to the message before continuing. Because message-based communication is inherently asynchronous, it may not be appropriate in such situations. However, most MOM systems have the means to group request and response as a single pseudo-synchronous transaction.

Historically, message queues have used their own closed protocols, limiting the ability of different operating systems or programming languages to interact in a heterogeneous set of environments. Over time, three standards have emerged that are used in open source message queue implementations: Advanced Message Queuing Protocol (AMQP), Text Messaging Protocol (STOMP), and MQTT (formerly MQ Telemetry Transport). The most widespread protocol is AMQP, which was standardized by ISO [4].

Taking into account all the above advantages and disadvantages, it can be argued that the software based on messages is much better than other analogues, but increases the complexity of the system. The functionality of this software depends on the message queue used and the message broker.

REFERENCES

1. Curry, Edward. (2005). Message-Oriented Middleware. http://www.edwardcurry.org/publications/curry_MfC_MOM_04.pdf
2. Oracle. 2021. Message-Oriented Middleware (MOM). https://docs.oracle.com/cd/E19340-01/820-6424/aeraq/index.html
3. Edward Curry. 2011. Extending Message-Oriented Middleware using Inthttps://web.archive.org/web/20110726015301/http://www.edwardcurry.org/web_publications/curry_DEBS_04.pdferception.
4. ISO. 2021. ISO/IEC 19464:2014 Information technology – Advanced Message Queuing Protocol (AMQP) v1.0 specification.: https://www.iso.org/standard/64955.html