Message-oriented Middleware (MOM)

In this lesson, we'll discuss message-oriented middleware in a bit more depth.

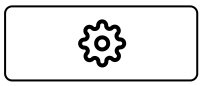
We'll cover the following

- Microservices & MOMs
- Variants of MOMs
  - Java messaging service
  - Advanced message queuing protocol
  - ZeroMQ
  - MQTT

Microservices & MOMs

Microservices are decoupled by a MOM. A microservice sends a message to or receives it from the MOM. This means that **the sender and the recipient do not know each other**, only the communication channel. Service discovery is therefore not necessary. Sender and recipient find each other via the topic or queue through which they exchange messages.

Load balancing is also easy. If several recipients have registered for the same communication channel, a message can be processed by one of the recipients and the load can be distributed, thereby eliminating the need for a specific infrastructure for load balancing.



However, a MOM is **a complex software that handles all communication**. Therefore, the MOM must be highly available and has to offer a high throughput. MOMs are generally very mature products, but ensuring adequate performance under all conditions requires a lot of know-how, for example, concerning the configuration.

Variants of MOMs#

In the area of MOMs, the following products are popular.

Java messaging service#

JMS (https://jcp.org/aboutJava/communityprocess/final/jsr914/index.html) (Java Messaging Service) is a standardized API for the programming language Java and part of the Java EE standard.

Well known implementations are Apache ActiveMQ (http://activemq.apache.org/) or IBM MQ (http://www-03.ibm.com/software/products/en/ibm-mq), which was previously known as IBM MQSeries. However, many more JMS products (https://en.wikipedia.org/wiki/Java_Message_Service#Provider_implementations) are available. JMS implementations might also provide APIs for other programming languages than Java. Those would not be covered by the JMS specification as it is specific for Java. Java application servers that support the entire Java EE profile – not just the web profile – have to contain a JMS implementation, so that JMS is often anyway available.

Advanced message queuing protocol#

AMQP (https://www.amqp.org/) (Advanced Message Queuing Protocol) does not standardize an API, but a network protocol at the level of TCP/IP. This allows for a simpler exchange of the implementation.

RabbitMQ (https://www.rabbitmq.com/), Apache ActiveMQ (http://activemq.apache.org/), and Apache Qpid (https://qpid.apache.org/) are the best known implementations of the AMQP standard. There are also a lot more implementations (https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol#Implementations).

ZeroMQ#

In addition, there is ZeroMQ (http://zeromq.org/), which does not comply with any of the standards. ZeroMQ is a library i.e. there is no need to install a message broker to use it.

MQTT#

Lastly, MQTT (http://mqtt.org/) is a messaging protocol that plays a prominent role for the Internet of Things (IoT).

All of these MOM technologies can be used to build a microservices system. If a certain technology is already in use and knowledge about its use is readily available, a decision to use an already known technology can make a lot of sense. It takes a lot of effort to run a microservices system.

The **use of a well-known technology reduces risk and effort**. The requirements for availability and scalability of MOMs are high. A well-known MOM can help to meet these requirements in a simple way.

QUIZ

1 When a MOM is used with a microservices system, the microservices communicate with _____.

A) each other

B) the MOM

C) the event queue

Submit Answer

Question 1 of 3
0 attempted

Reset Quiz

In the next lesson, we'll discuss the architecture of Kafka.

← Back

Introduction

Next →

The Architecture of Kafka

Mark as Completed

Report an Issue