Enterprise Computing

—

Cost-Effectiveness of Apache ActiveMQ and Amazon MQ

Muhammad Taha,
Musadiq Raees,
Muhammad Nouman Shahzad,
Christian Hildebrandt

January 11, 2018

ActiveMQ and Amazon MQ

Modern software systems often are composed of many different components. Instead of being designed as one big, monolithic system nowadays software tends to be divided in many smaller, independent parts. These parts are often even written in different programming languages, depending on its task and the environment it is operating in. For coordination of and communication between these components a messaging system can be of use – if not even necessary. Apache ActiveMQ and Amazon MQ are perfect for this job. They provide a message broker for delivering customized messages between many endpoints, written in a variety of programming languages. Two different modes, or architectures, can be used. On one hand there is direct peer to peer communication, where one system endpoint is sending messages directly to another endpoint. Or on the other hand a Publisher-Subscriber pattern can be used, meaning that many subscribers will be notified of new messages coming from a publisher they subscribed to simultaneously. A broker object acts as the middle man in this kind of messaging architecture.

It is important to mention that Amazon MQ is merely a system that is built on top of ActiveMQ. It provides the user with a higher level of abstraction while adding some degree of convenience and options for evaluation. Being one of many of Amazon's Web Services it manages the setup (meaning installing and updating), migration, and availability of the ActiveMQ broker objects. In addition to that it provides metrics that can be viewed and analyzed with CloudWatch, Amazon's cloud and network monitoring service.

Exercise Roadmap

Since both of the systems, ActiveMQ and Amazon MQ, were new to all our group members, we decided to first get acquainted with them in more detail. We planned to try out ActiveMQ for ourselves, see how to set it up from scratch, write some basic messaging code, and get a fundamental feeling for the features it provides. After that the focus was to shift to Amazon MQ. We planned to find out where the differences between the two systems lie and, more importantly, what Amazon MQ provides on top of ActiveMQ. Additionally we wanted to find and review already existing benchmarking tools for the two systems. This was also supposed to help us identify notable and relevant metrics that could give some indication about the cost-effectiveness of the systems. The definition and implementation of concrete benchmark tests was scheduled as the next step. Finally, the last step of the exercise roadmap was to review and reprocess our results and bring them in a presentable form, e.g. design the presentation poster.

Cost-Effectiveness

It has not been easy for us to define, what the term 'cost-effectiveness' exactly means – especially in the context of comparing ActiveMQ with Amazon MQ. ActiveMQ is open source and thus free to use for everyone. Amazon however charges the user of their cloud services on an 'per hour' basis. So the question which of the two is less expensive – thus being more "cost"-effective – is trivial and ActiveMQ is the obvious winner.

We therefore thought of other forms of cost-effectiveness. What about execution times and performance? Or the time it takes to setup and manage the systems? Does the amount of lines of code you have to write provide information about cost-effectiveness? There are many different ways to compare the two systems. The results of our benchmarks were supposed to be repeatable and provide some degree of measurable difference between the two systems. This is why in the end we decided to...?

Our Benchmarks

- defined benchmarks

Results

- exciting results with lots of plots and graphs, whoohoo!