Throughput Performance of Popular JMS Servers

Michael Menth, Robert Henjes, Christian Zepfel, and Sebastian Gehrsitz
Dept. of Computer Science, University of Wuerzburg
Wuerzburg, Germany
{menth,henjes,gehrsitz,zepfel}@informatik.uni-wuerzburg.de

ABSTRACT

The Java Messaging Service (JMS) facilitates communication among distributed software components according to the publish/subscribe principle. If the subscribers install filter rules on the JMS server, JMS can be used as a message routing platform, but it is not clear whether its message throughput is sufficiently high to support large-scale systems. In this paper, we investigate the capacity of three high performance JMS server implementations: FioranoMQ, SunMQ, and WebsphereMQ. In contrast to other studies, we focus on the message throughput in the presence of filters and show that filtering reduces the performance significantly. We present models for the message processing time of each server and validate them by measurement.

Categories and Subject Descriptors: D.2.8 [Software Engineering]: Metrics–*[performance measures]*

General Terms: Measurement, Performance

Keywords: Publish/Subscribe, Server Performance, Java Messaging Service

1. THE JAVA MESSAGING SERVICE

Messaging facilitates the communication between remote software components. The Java Messaging Service (JMS) standardizes this message exchange. The so-called publishers generate and send messages to the JMS server, the so-called subscribers consume these messages – or a subset thereof – from the JMS server, and the JMS server acts as a relay node, which controls the message flow by various message filtering options. This is depicted in Figure 1. Publishers and subscribers rely on the JMS API and the JMS server decouples them by acting as an isolating element. As a consequence, publishers and subscribers do not need to know each other.

The JMS offers several modes. In the persistent mode, messages are delivered reliably and in order. In the durable mode, messages are also forwarded to subscribers that are currently not connected while in the non-durable mode, messages are forwarded only to subscribers who are presently online. Thus, the server requires a significant amount of buffer space to store messages in the durable mode. In this study, we consider the persistent but non-durable mode if not mentioned differently.

Information providers with similar themes may be grouped together and publish to a so-called common topic; only those subscribers having subscribed for that specific topic receive their messages. Thus, topics virtually separate the JMS server into several logical sub-servers. Topics provide only a very coarse and static method for message selection. In addition, topics need to be configured on the JMS server before system start. Filters are another option for message selection. A subscriber may install a message filter on the JMS server, which effects that only the messages matching the filter rules are forwarded instead of all messages in the corresponding topic. In contrast to topics, filters are installed dynamically during the operation of the server. Figure 2 shows that a JMS message consists of three parts: the message header, a user defined property header section, and the message payload itself.

This work was funded by Siemens AG, Munich. The authors alone are responsible for the content of the paper.

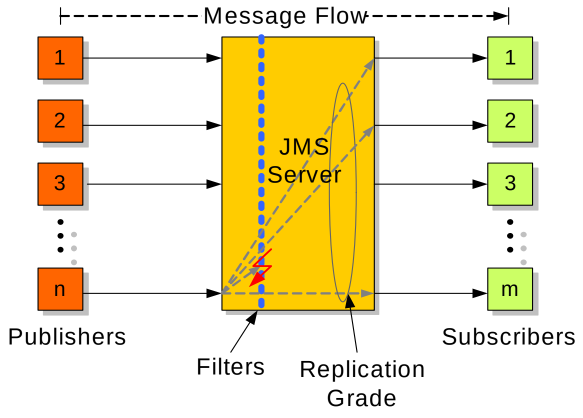

Figure 1: The JMS server decouples publishers and subscribers.

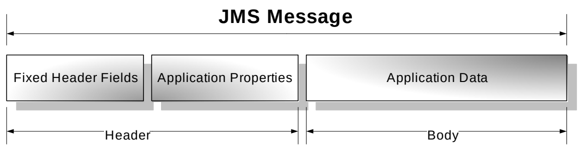

Figure 2: The JMS message structure.

So-called correlation IDs are ordinary 128 byte strings that can be set in the header of JMS messages. Correlation ID filters try to match these IDs whereby wildcard filtering is possible, e.g., in the form of ranges like [#7;#13]. Several application-specific properties may be set in the property section of the JMS message. Application property filters try to match these properties. Unlike to correlation ID filters, a combination of different properties may be specified which leads to more complex filters with a finer granularity. After all, topics, correlation ID filtering, and application property filtering are three different possibilities for message selection with different semantic granularity and different computational effort.

2. RESULTS

We first describe some general findings from our experiments qualitatively and present then quantitative performance models for the maximum message throughput of each investigated server type. We consider both the throughput of messages received by the server as well as the overall rate of messages received and dispatched by the server.

2.1 General Findings

(1) The throughput of the three investigated server types spans over several orders of magnitude with FioranoMQ achieving the highest one, followed by SunMQ, and WebsphereMQ achieving the lowest one.

(2) The throughput is significantly larger in the non-persistent mode than in the persistent mode. The difference depends on the server type.

(3) A message may be replicated and forwarded to r different subscribers. Then, we call r the replication grade of a message. This average replication grade reduces the received throughput and increases the overall throughput of the server.

(4) The throughput depends also the filter installations. FioranoMQ can handle simple correlation ID filters more efficiently than application property filters while both filter types requires the same effort for SunMQ and WebsphereMQ.

(5) The message throughput is limited either by the processing logic for small messages or by the transmission capacity for large messages.

(6) The number of configured topics hardly affects the overall throughput of the server.

(7) Complex OR-filters allow a larger message throughput than an equivalent number of simple filters. The performance gain depends significantly on the server type.

(8) The complexity of AND-filters reduces the message throughput for FioranoMQ and SunMQ. The order of the filter components matters. This can be used to optimize the formulation of filter rules. In contrast, WebsphereMQ requires the same time to process a message regardless of the filter complexity and the order of the filter components.

2.2 Performance Models

We conducted extensive and complex experiments to assess the dependencies of the maximum message throughput on various parameters. We propose analytical models to characterize the message processing time which is the inverse of the received message throughput. We fit the model parameters such that the formulae approximate well the experimental throughput.

2.2.1 Performance Model for FioranoMQ

The message processing time depends both on the number of filters n_{fltr} installed on the server and the message replication grade r. A very simple model is already appropriate to characterize the message processing time B:

$$B = t_{rcv} + n_{fltr} \cdot t_{fltr} + r \cdot t_{tx}. \qquad (1)$$

The parameter t_{rcv} is a fixed time overhead for each received message. The time to check a single filter is t_{fltr}, and the filtering effort increases linearly with the number n_{fltr} of installed filters. Finally, t_{tx} describes the time to dispatch and to send a single message for a matching filter. The parameter values $t_{rcv} = 8.52 \cdot 10^{-7}$ s, $t_{fltr} = 7.02 \cdot 10^{-6}$ s, and $t_{tx} = 1.70 \cdot 10^{-5}$ s describe the server behavior for correlation ID filters and $t_{rcv} = 4.10 \cdot 10^{-6}$ s, $t_{fltr} = 1.46 \cdot 10^{-5}$ s, and $t_{tx} = 1.62 \cdot 10^{-5}$ s describe it for application property filters.

2.2.2 Performance Model for SunMQ

The message processing time depends on the number of all filters n_{fltr}^{all}, the number of different filters n_{fltr}^{diff}, and the replication grade r. The message processing time B can be approximated by

$$B = t_{rcv} + n_{fltr}^{all} \cdot t_{fltr}^{all} + n_{fltr}^{diff} \cdot t_{fltr}^{diff} + r \cdot t_{tx}. \qquad (2)$$

The meaning of the parameters t_{rcv} and t_{tx} is like above. The processing time for filters is more complex than above, probably due to internal optimization. The filtering effort increases linearly with the number of all filters n_{fltr}^{all} and the time to check a single filter is t_{fltr}^{all}. The number of different filters imposes an extra overhead of $n_{fltr}^{diff} \cdot t_{fltr}^{diff}$. The parameter values $t_{rcv} = 1.14 \cdot 10^{-4}$ s, $t_{fltr}^{all} = 2.10 \cdot 10^{-6}$ s, $t_{fltr}^{diff} = 2.12 \cdot 10^{-6}$ s, and $t_{tx} = 4.01 \cdot 10^{-5}$ s characterize well the message processing time for the SunMQ both for correlation ID and application property filtering.

2.2.3 Performance Model for WebsphereMQ

The message processing time depends only on the number of filters n_{fltr}. In contrast to FioranoMQ and SunMQ, it does not depend on the replication grade r. Thus, the time to send messages is obviously so small that it is not noticeable for a replication grade of up to $r = 40$. The following model approximates well the message processing time B:

$$B = t_{rcv} + n_{fltr} \cdot \sqrt{(n_{fltr})} \cdot t_{fltr} \qquad (3)$$

The parameter values $t_{rcv} = 7.03 \cdot 10^{-4}$ s and $t_{fltr} = 1.10 \cdot 10^{-5}$ s approximate well the throughput for both correlation ID and application property filtering. A linear model like for the FioranoMQ or the SunMQ does not work for the approximation of the measurement results.

3. CONCLUSION

In this work, we have compared the message throughput of the FioranoMQ, SunMQ, and WebsphereMQ Java messaging system (JMS) server under various conditions. We developed analytical models to describe the maximum message throughput of the server depending on the average message replication grade r, the overall number of installed filters n_{fltr}^{all}, and the number of different filters n_{fltr}^{diff}. The throughput of all three server types is significantly different and the models also reveal a different scaling behavior with respect to the above parameters. The models can be used in practice to estimate whether a special JMS server suffices to handle the message rate in a certain application scenario. However, the parameter values mentioned in this paper are only valid for the hardware used in our experiments which are documented in [1].

4. REFERENCES

[1] M. Menth, R. Henjes, S. Gehrsitz, and C. Zepfel, “Throughput Comparison of Professional JMS Servers,” University of Würzburg, Institute of Computer Science, Technical Report, No. 380, Mar. 2006.