Benchmarking of Message-Oriented Middleware

Kai Sachs
TU Darmstadt, Germany
sachs@dvs.tu-darmstadt.de

Samuel Kounev
FZI Karlsruhe, Germany
skounev@acm.com

Stefan Appel,
Alejandro Buchmann
TU Darmstadt, Germany
lastname@dvs.tu-darmstadt.de

1. INTRODUCTION

Message-oriented middleware (MOM) is increasingly used as enabling technology for modern event-driven applications typically based on publish/subscribe (pub/sub) communication [1]. Many of these applications are designed for maximum scalability and flexibility and as such, they pose some serious performance issues for the underlying pub/sub middleware. Additionally, software designers face a new challenge: designing message-based communication flows which rely on asynchronous decoupled communication patterns. In order to develop good designs, system designers have to understand quality of service aspects and their performance costs. We believe that benchmarks are very helpful tools to analyse these aspects.

In this poster, we provide an overview of our past and current research in the area of MOM performance benchmarks. Our main research motivation is a) to gain a better understanding of the performance of MOM, b) to show how to use benchmarks for the evaluation of performance aspects and c)to establish performance modeling techniques. For a better understanding we first introduce the Java Message Service (JMS) standard. Afterwards, we provide an overview of our work on MOM benchmark development, i.e., we present the *SPECjms2007* benchmark [4] and the brand new *jms2009-PS* [3], a test harness designed specifically for JMS-based pub/sub. We outline a new case study with jms2009-PS and present first results of our work-in-progress. Due to space constraints we refer the interested reader for related work to [4] (for an overview in the area of MOM) and to [2] (for performance of event-based systems in general).

2. JAVA MESSAGE SERVICE

The JMS interface [6] is supported by all major MOM vendors and has established itself as the de facto industry standard interface for MOM. It supports two messaging models: *point-to-point (P2P)* and *publish/subscribe (pub/sub)*. P2P messaging is built around the concept of message *queues* where each queue forms a virtual communication channel. Each incoming message is processed by a single consumer (1:1). In contrast, pub/sub messaging is built around the concept of *topics*. Each message may be delivered to multiple consumers which are interested in a specific topic (1:n). JMS queues and topics are commonly referred to as *destinations*. Furthermore, JMS offers so-called selectors. Selectors allow consumers to define filters for incoming messages in SQL-like notation.

DEBS'09, July 6-9, Nashville, TN, USA.

ID	JMS Sever	JVM	DB	Max
A	ActiveMQ 4.1.2	Sun, Java 6.0	Derby	720
B	ActiveMQ 4.1.2	Sun, Java 6.0	MySQL	770
C	ActiveMQ 4.1.2	Oracle JRockit, Java 6.0	Derby	830
D	ActiveMQ 4.1.2	Oracle JRockit, Java 6.0	MySQL	870
E	*Comm. Vendor*	Oracle JRockit, Java 5.0	*File*	710

Table 1: Configurations and Vertical Results

3. BENCHMARKS

3.1 SPECjms2007 - A JMS Benchmark

SPECjms2007 is the first industry standard benchmark for JMS. It was developed by the Standard Performance Evaluation Corporation (SPEC) under the leadership of TU Darmstadt. The underlying application scenario models a supermarket's supply chain where RFID technology is used to track the flow of goods between different parties. The workload of this scenario is based on the experience of SPEC member organisations including IBM, Sun and Oracle. Seven interactions such as order management are modeled in detail to stress different aspects of MOM performance.

SPECjms2007 is implemented as a Java application framework comprising multiple Java Virtual Machines and threads distributed across a set of client nodes [5]. A detailed workload description and a case study using SPECjms2007 is provided in [4]. As an example for SPECjms2007 usage, we present performance results for ActiveMQ [7], an open source JMS implementation, with different configurations (see Table 1 and Figure 1).[1]

[1]SPECjms2007 is a trademark of the Standard Performance Evaluation Corporation(SPEC). The results or findings in this publication have not been reviewed or accepted by SPEC, therefore no comparison nor performance inference can be made against any published SPEC result. The official web site for SPECjms2007 is located at http://www.spec.org/osg/jms2007.

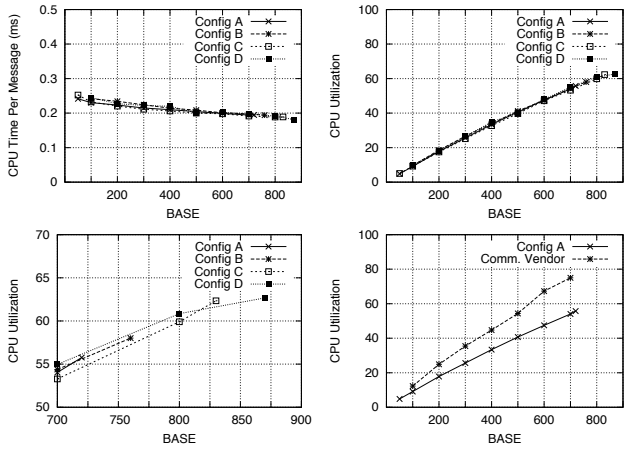

Figure 1: SPECjms2007 Results for ActiveMQ

3.2 jms2009-PS - A pub/sub Benchmark

jms2009-PS is a novel benchmark for pub/sub systems [3]. jms2009-PS is built on top of the SPECjms2007 framework and workload [4] using pub/sub communication for each interaction. jms2009-PS offers two different ways to achieve this: a) using one topic per interaction or b) using a separate topic for each message type in an interaction. Additionally, queues can be used instead of topics for P2P communication. In contrast to other frameworks, jms2009-PS allows to define complex traffic scenarios with different destinations, message types (using different sizes), service levels and filters.

4. A CASE STUDY USING JMS2009-PS

We present initial results of our work-in-progress: a case study using jms2009-PS. Our motivation for this case study is to analyse different performance aspects of MOM. We focus on pub/sub communication as well as on the influence of design decisions on system performance. System developers have to decide at design time what message communication patterns to use, e.g. destination types, message filter complexity, and number of destinations. Such decisions strongly influence scalability, flexibility, and performance, whereas wrong decisions can cause serious problems and high follow-up costs for correcting them afterwards. Therefore, we see a strong need for methods and tools, that allow system designers to predict the impact of their decisions in advance. One class of such tools are benchmarks, which are highly useful for performance evaluation of complex system architectures. In our case study we use the latest version of jms2009-PS which contains specific extensions to meet our requirements.

One focus of our case study is to evaluate the choice of destination types, i.e. queues vs. topics. In case of multiple consumers per message, it is easy to see that topics are the better option. However, for message exchanges with a single consumer (P2P), the developer may choose either topics or queues. One reason for using topics can be better scalability and flexibility compared to queues.

For example, consider a system where incoming orders are consumed by an order management component. We extend the system by a new controlling component that also receives all incoming order messages, i.e., changing the communication pattern from 1:1 to 1:n. In case of an implementation using queues, at minimum the MOM has to be reconfigured (which can lead to downtime) and likely the code of the order management system would have to be adjusted and the component redeployed. Instead, in a solution based on topics, the new controller component only has to subscribe to the corresponding topics. Other software components are not affected and will not even notice the new component. However, what are the costs for this flexibility? Other important questions are, for example., related to the number of message destinations (e.g. one topic for each message type vs. one topic for all message types), transaction costs as well as overhead of durable subscriptions, number of subscribers, filter complexity and message sizes.

Since a system designer has to deal with such questions, we are evaluating these aspects in our case study by analysing various scenarios. We focus on pub/sub communication and compare it, where meaningful, with queues. In contrast to other case studies, we use a complex and comprehensive workload with different message types, high parallelism, high number of destinations, and different message sizes. Our server setup is comparable to a state-of-the-art real-world environment; as JMS implementation we use a major commercial product for our experiments.

5. SUMMARY

We gave a brief overview of our past and ongoing research work in the area of MOM benchmarks and presented the two benchmarks including results: SPECjms2007, a standardized JMS benchmark, and jms2009-PS, a pub/sub performance test harness. Furthermore, we outlined a case study in which we use jms2009-PS to evaluate different performance aspects of MOM.

6. REFERENCES

[1] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2):pages 114–131, 2003.

[2] S. Kounev and K. Sachs. Benchmarking and performance modeling of event-based systems. *Special issue of it - Information Technology on Complex Event Processing*, 2009. To appear.

[3] K. Sachs, S. Kounev, S. Appel, and A. Buchmann. A Performance Test Harness For Publish/Subscribe Middleware. In *SIGMETRICS/Performance 2009: Demo Competition*. SIGMETRICS, 2009.

[4] K. Sachs, S. Kounev, J. Bacon, and A. Buchmann. Performance evaluation of message-oriented middleware using the SPECjms2007 benchmark. *Performance Evaluation*, 66(8):410–434, Aug. 2009.

[5] K. Sachs, S. Kounev, M. Carter, and A. Buchmann. Designing a Workload Scenario for Benchmarking Message-Oriented Middleware. In *Proceedings of the 2007 SPEC Benchmark Workshop*, 2007.

[6] Sun Microsystems, Inc. Java Message Service (JMS) Specification - Version 1.1. Technical report, 2002.

[7] The Apache Software Foundation. ActiveMQ. http://activemq.apache.org/, 2009.