A Performance Test Harness For Publish/Subscribe 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

ABSTRACT

Publish/subscribe is becoming increasingly popular as communication paradigm for loosely-coupled message exchange. It is used as a building block in major new software architectures and technology domains such as Enterprise Service Bus, Enterprise Application Integration, Service-Oriented Architecture and Event-Driven Architecture. The growing adoption of these technologies leads to a strong need for benchmarks and performance evaluation tools in this area. In this demonstration, we present *jms2009-PS*, a benchmark for publish/subscribe middleware based on the Java Message Service standard interface.

1. INTRODUCTION

The publish/subscribe (pub/sub) [2] paradigm is nowadays used in many applications which are mostly designed for maximum scalability. Therefore, such applications pose some serious performance and scalability challenges for the underlying pub/sub middleware. In order to solve these issues, we see a strong need for performance tools and benchmarks targeting pub/sub middleware. To the best of our knowledge, currently no benchmarks or performance test harness exist specifically targeted at pub/sub systems. Some general guidelines for designing a benchmark suite for (distributed) pub/sub systems are presented in [1], however, no specific implementation is provided. A possible starting point for developing such a benchmark is the SPECjms2007 standard workload.

SPECjms2007 is the first industry-standard benchmark specialized for Message Oriented Middleware (MOM) [6]. It exercises messaging products using the JMS (Java Message Service) standard interface [7] and was developed by the Standard Performance Evaluation Corporation (SPEC) under the leadership of TU Darmstadt. It offers a comprehensive, standardized workload based on the experience of SPEC member organisations including IBM, Sun and Oracle. One of the major benefits of SPECjms2007 is that it not 'only' provides a benchmark, but also a complex performance test harness and a framework for performance analysis. It allows to create custom workload scenarios and interactions or to modify existing ones. Examples for such user-defined scenarios can be found in [3] and [4]. SPECjms2007 has received a lot of attention by both industry and academia and is used for software quality assurance by major MOM vendors. The goal of SPECjms2007, however, was not to develop a pub/sub test harness but rather to provide a general MOM benchmark. As such, the benchmark makes extensive use of the Point-to-Point (PtP) communication via queues which dominates the overall system workload [5]. In addition, it does not use JMS selectors.

To meet the growing demand for benchmarking pub/sub middleware, we developed a performance harness for pub/sub using the SPECjms2007 framework as a basis. In this demonstration we present our benchmark called *jms2009-PS*. We kept the business logic of SPECjms2007, but instead of using PtP communication, we applied pub/sub for all kinds of message exchange making heavy use of selectors. Furthermore, we extended the workload scenario by incorporating additional flexibility to customize workloads.

2. SPECJMS2007

2.1 Workload Scenario

The application scenario models a supermarket's supply chain where RFID technology is used to track the flow of goods. The participants involved are the supermarket's company headquarters (HQ), its stores (SMs), its distribution centers (DCs) and its suppliers (SPs). SPECjms2007 defines seven interactions between the participants in a supermarket supply chain:

1. Order/shipment handling between SM and DC
2. Order/shipment handling between DC and SP
3. Price updates sent from HQ to SMs
4. Inventory management inside SMs
5. Sales statistics sent from SMs to HQ
6. New product announcements sent from HQ to SMs
7. Credit card hot lists sent from HQ to SMs

While interactions 1 and 2 represent a chain of messages, the other interactions include a single message exchange [4]. The workload specification defines 19 different message types and their properties ((non-)transactional, (non-)persistent, etc.) as well as a list of queues and topics.

2.2 Framework

SPECjms2007 is implemented as a Java application comprising multiple JVMs and threads distributed across a set of *client nodes*. For every destination (queue or topic), there is a separate Java class called *Event Handler (EH)* that encapsulates the application logic executed to process messages sent to this destination. Event handlers register as listeners for the queue/topic and receive call backs from the messaging infrastructure as new messages arrive. In addition, for every physical location, a set of threads (referred to as *driver threads*) is launched to drive the benchmark interactions that are logically started at this location.

An important component of the framework is the measurement functionality. Event handlers measure the delivery time of each incoming message, the total number of received/sent messages by source/target and their execution time. After a specified interval and at the end of the experiment, *controller threads* collect the measured data and prepare it for further processing. These measurements are used for runtime statistics and, later on, for reports at different granularity. Additionally, a postauditor checks the measurement results based on a set of rules (e.g. all messages were received) to validate the correctness of a test run. Further features are a preauditor and a prediction component which calculates the expected throughput.

3. JMS2009-PS - A PUB/SUB BENCHMARK

In this demonstration we present a novel benchmark for pub/sub systems: *jms2009-PS*. We built jms2009-PS on top of the SPECjms2007 framework and implemented the workload using pub/sub communication for each interaction in two different ways: a) using one topic per interaction and b) using one topic per step (message type) in an interaction. The two implementations are identical concerning the total number of messages/traffic and subscriptions. However, they differ in four important points:

1. number of topics,
2. traffic per topic,
3. complexity of filter statements and
4. number of subscribers per topic.

In our first implementation, all types of messages are exchanged using one common topic per interaction. Each message consumer (e.g. order department in DC 1) subscribes to this topic using two different filter values which define the messages he is interested in: message type (e.g. orders) and their own location ID (e.g. DC 1). To be able to identify the corresponding message consumer the middleware requires that the message producers have previously assigned these two properties (message type, location) to each incoming message.

Our second implementation assigns a specific topic to each type of message (e.g. one topic for orders, one for invoices). Consequently, message consumers do not have to specify the message type of their interest at subscription time, but only their location ID. It is easy to see that a) the number of subscribers per topic decreases and b) the filtering is simpler (only one property to check) in the second implementation compared to the first one. Therefore, the two implementations stress the system in various ways and allow to evaluate different performance aspects. The user can decide for each interaction which implementation to use. Additionally, we extended the existing workload by adding new features. For example, a user can decide for each step in an interaction whether a persistent or non-persistent message is sent as well as whether subscriptions are durable.

4. DEMONSTRATION

Our demonstration shows how to use jms2009-PS for performance evaluation of pub/sub middleware and provides an introduction to the underlying SPECjms2007 framework. We explain how to create a custom workload scenario and illustrate how to setup and run the benchmark. Finally, we have a closer look at the result reports and present some experimental results.

5. ONLINE RESOURCES

A detailed description of SPECjms2007 including a user's guide and a design document is available at [6]. A comprehensive description of the workload and a case study using customized scenarios can be found in [4]. A document describing the new features of jms2009-PS is under way and will be available soon at the authors' website.

6. CONCLUSIONS

jms2009-PS is a powerful tool to evaluate pub/sub middleware using a comprehensive workload. Given that jms2009-PS is built on top of SPECjms2007, it uses the JMS standard interface. It allows to analyse different features of pub/sub middleware and their influence on the system's throughput and performance. jms2009-PS is a major contribution in the area of pub/sub benchmarking and performance engineering of event-based systems. Our future research will focus on case studies using jms2009-PS as well as the development of a benchmark for *distributed* event-based systems.

7. REFERENCES

[1] A. Carzaniga and A. L. Wolf. A Benchmark Suite for Distributed Publish/Subscribe Systems. Technical report, Department of Computer Science, University of Colorado, 2002.

[2] 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.

[3] J. Happe, H. Friedrich, S. Becker, and R. H. Reussner. A pattern-based Performance Completion for Message-oriented Middleware. In *Proc. of the 7th International Workshop on Software and Performance (WOSP)*, pages 165–176. ACM, 2008.

[4] K. Sachs, S. Kounev, J. Bacon, and A. Buchmann. Performance evaluation of message-oriented middleware using the SPECjms2007 benchmark. *Performance Evaluation*, 2009. To appear, online available via doi:10.1016/j.peva.2009.01.003.

[5] K. Sachs, S. Kounev, and A. Buchmann. Performance Modeling of Message-Oriented Middleware - A Case Study. 2009. In review.

[6] Standard Performance Evaluation Corporation. SPECjms2007. http://www.spec.org/jms2007/, 2009.

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