Benchmarking and Performance Modeling of Event-Based Systems

Modellierung und Bewertung von Ereignis-basierten Systemen

Samuel Kounev, Universität Karlsruhe (TH),
Kai Sachs, TU Darmstadt

Summary   Event-based systems are used increasingly often to build loosely-coupled distributed applications. With their growing popularity and gradual adoption in mission critical areas, the need for novel techniques for benchmarking and performance modeling of event-based systems is increasing. In this article, we provide an overview of the state-of-the-art in this area considering both centralized systems based on message-oriented middleware as well as large-scale distributed publish/subscribe systems. We consider a number of specific techniques for benchmarking and performance modeling, discuss their advantages and disadvantages, and provide references for further information. The techniques we review help to ensure that systems are designed and sized to meet their quality-of-service requirements. ►►►
Zusammenfassung   Ereignis-basierte Systeme werden immer häufiger beim Aufbau von hochverteilten Systemen und als

Kommunikationstechnologie in Enterprise-Software eingesetzt. Durch den damit einhergehenden Bedeutungszuwachs treten Methodiken zur Bewertung und Vorhersage von Leistungsmerkmalen und Servicequalität wie Benchmarks und Performance Modellierungsansätze immer mehr in den Vordergrund. In diesem Artikel geben wir einen Überblick über den aktuellen Stand der Forschung im Bereich Performance-Modellierung und Benchmarking von ereignis-basierten Systemen. Hierzu betrachten wir zwei verschiedene Ansätze: zentralistische Infrastrukturen (message-oriented middleware) und verteilte publish/subscribe Systeme. Dabei beleuchten wir Vor- und Nachteile der bestehenden Benchmark- und Modellierungsansätze und gehen darauf ein, wie diese eingesetzt werden können, um mögliche Engpässe und Servicequalitätsmerkmale vorherzusagen und eine entsprechende Kapazitätsplanung vorzunehmen.

1 Introduction

Event-based systems (EBS) have been gaining increasing attention in many industry domains including manufacturing, transportation, health-care and supply chain management. Moreover, the publish-subscribe paradigm is now used as a building block in major new software architectures and technology domains such as enterprise service bus (ESB), enterprise application integration (EAI), service-oriented architecture (SOA) and event-driven architecture (EDA).

With the advent of ambient intelligence and ubiquitous computing, many new applications of EBS have been proposed [1]. Novel event-based applications, however, pose some serious performance and scalability challenges.

To avoid the pitfalls of inadequate Quality-of-Service (QoS), it is essential that EBS are subjected to a rigorous performance and scalability analysis before they are put into production. Common performance metrics of interest are the expected event notification latency as well as the utilization and message throughput of the various system components (e. g., event brokers, network links). Obtaining such information is essential in order to determine the optimal system topology, configuration and capacity that would provide adequate QoS to applications at a reasonable cost.

In this article, we provide an overview of the major methods and techniques for benchmarking and performance modeling of EBS. We first consider centralized systems based on Message-Oriented Middleware (MOM). We provide an overview of current benchmarks for MOM focusing on the SPECjms2007 industry-standard benchmark. We then review several techniques for performance modeling of MOM systems. We focus on a novel technique that we have developed which we present in more detail and illustrate by means of a case study. Following this, we turn to large-scale distributed event-based systems (DEBS). We survey the state-of-the-art in performance modeling of DEBS considering both analytical and simulation-based approaches. We discuss the advantages and disadvantages of different approaches and provide references to several practical case studies that illustrate them. The modeling techniques we review help to identify and eliminate system bottlenecks and ensure that systems are designed and sized to meet their QoS requirements.

2   MOM Systems

The most popular MOM platforms currently used in industry include IBM WebSphere MQ, TIBCO Enterprise Message Service, Progress SonicMQ, Oracle WebLogic Server, Sun Java System MQ, Apache ActiveMQ and FioranoMQ. Most of these platforms support the JMS (Java Message Service) standard interface for accessing MOM services. JMS supports two communication modes: *point-to-point* and *publish/subscribe*. The former is used for one-to-one communication through message queues managed by the MOM server. The latter provides a topic-based publish/subscribe service with limited content-based filtering capabilities.

2.1 Benchmarks

Over the last decade several proprietary and open-source benchmarks for evaluating MOM platforms have been developed and used in the industry including SonicMQ' Test Harness [2], IBM's Performance Harness for Java Message Service [6], Apache's ActiveMQ JMeter Performance Test [5] and JBoss' Messaging Performance Framework [3]. Benchmarks not only help to compare alternative platforms and validate them, but can also be exploited to study the effect of different platform configuration parameters on the overall system performance. While the benchmarks we mentioned have been employed extensively for performance testing and product comparisons, unfortunately, they use artificial workloads that do not reflect any real-world application scenario. Furthermore, they typically concentrate on stressing individual MOM features in isolation and do not provide a comprehensive and representative workload for evaluating the overall MOM server performance.

To address these concerns, in September 2005, we launched a project at the Standard Performance Evaluation Corporation (SPEC) with the goal to develop a standard benchmark for evaluating the performance

and scalability of MOM products. The effort continued over a period of two years and the new benchmark was released at the end of 2007. The benchmark was called SPECjms2007 and it was developed with the participation of TU Darmstadt, IBM, Sun, BEA, Sybase, Apache, Oracle and JBoss. SPECjms2007 exercises messaging products through the JMS standard interface which is supported by all major MOM vendors.

SPECjms2007 is based on a novel application in the supply chain management domain that comprises a set of supply chain interactions between a supermarket company, its stores, its distribution centers and its suppliers. In addition to providing a standard workload and metrics for MOM performance, the benchmark allows users to customize the workload to their needs by configuring it to stress selected features of the MOM infrastructure in a way that resembles a given target customer workload. Thus, the benchmark provides a flexible and robust tool for in-depth performance evaluation of MOM servers. In [7], we presented a comprehensive workload characterization of the SPECjms2007 workload capturing the information needed in order to customize the workload. Building on the workload characterization, we developed a novel methodology for performance evaluation of MOM platforms. The methodology is the first one that uses a standard benchmark providing both a representative workload as well as the ability to tailor the workload to the user's requirements.

Using SPECjms2007 as a basis, we developed a benchmark called *jms2009-PS* specifically targeted at publish/subscribe systems [8]. jms2009-PS is built on top of the SPECjms2007 framework and workload using publish/subscribe communication for all of the seven interactions. It is freely configurable and allows to define complex traffic scenarios with different destinations (queues and topics), message types (using different sizes), service levels and filters.

Case Study

In the following, we present a case study with SPECjms2007 showing how the benchmark can be used to evaluate alternative system configurations. We use SPECjms2007 to analyse the influence of the Java Virtual Machine (JVM) and the persistence store on the total throughput of the open source JMS Server ActiveMQ 4.1.2. As a persistence store, we use Derby and MySQL. We configured the system using two popular commercial Java 6.0 JVMs hereafter referred to as Vendor A and Vendor B. Overall, we considered the following four system configurations:

A Vendor A – Derby
B Vendor A – MySQL
C Vendor B – Derby
D Vendor B – MySQL

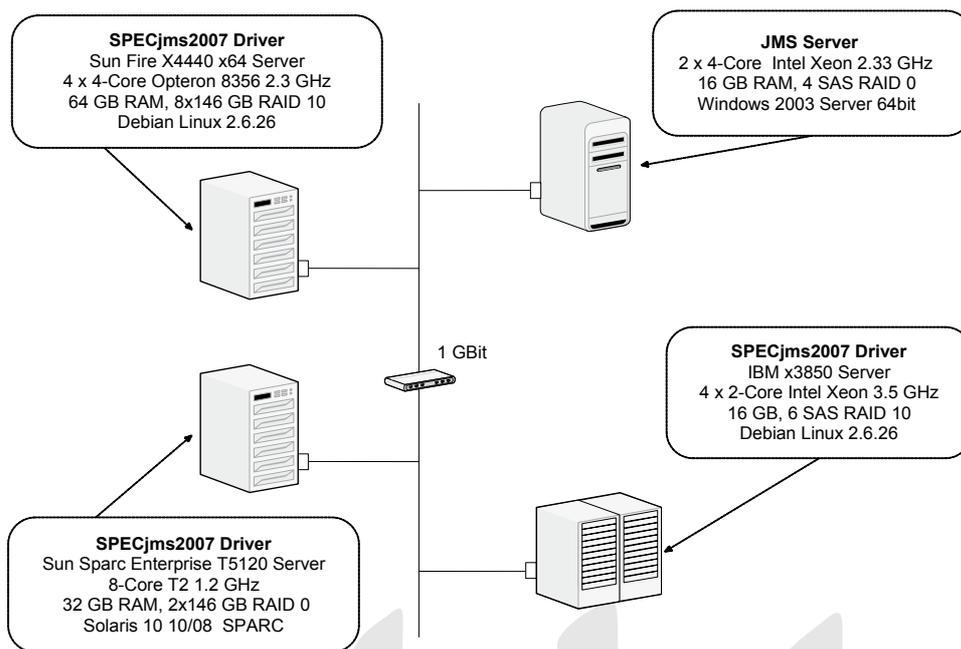The hardware setup is illustrated in Fig. 1. We ran the benchmark with all four configurations and measured

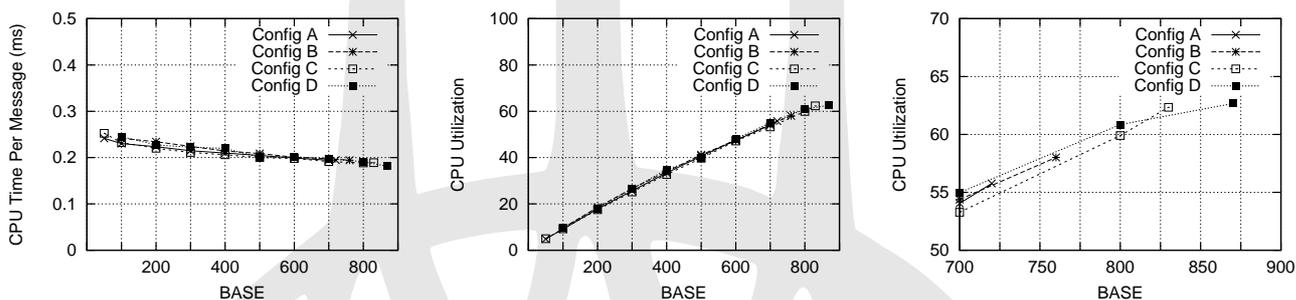**Figure 1** SPECjms2007 experimental environment.

Figure 2 Measured CPU time per message and total server CPU utilization.

the maximum system throughput for each of them. Due to space restrictions here we only provide a summary of the most important findings. From the results, shown on Fig. 2 and Table 2, we observed that under comparable load the CPU utilization seems to be independent of the selected JVM and the persistence store. The CPU time per message as well as the total system utilization is similar for all four configurations.

Furthermore, the results show that the maximal system throughput is highly correlated with the chosen JVM and database (Table 1). For both JVMs, the configuration using MySQL as persistence store achieved higher throughput than the configuration using Derby (between 4.9% and 6.9%). In our experiments, the choice of the JVM had a higher impact on the system performance than the choice of the persistence store (up to 15.3%). The 'worst' configuration for vendor B (Config. C, Derby) outperforms the best configuration (Config. B, MySQL) of vendor A by 7.8%. Therefore, we conclude that the choice of the JVM is at least as important for maximizing the system throughput as the choice of the persistence store.

Another more detailed case study was published in [7]. There we consider a leading JMS platform, the Oracle WebLogic server, conducting an in-depth performance analysis of the platform under a number of different workload and configuration scenarios. We evaluate the server performance for both the point-to-point and publish/subscribe messaging domains studying the effect of individual workload characteristics on the server CPU utilization, the message throughput, the CPU processing time per message/KByte payload and the message delivery latency.

Table 1 Maximum system throughput.

| | | Java VM | |
		Vendor A	Vendor B
D	Derby	Config. A 720	Config. C 830
B	MySQL	770 Config. B	870 Config. D

Table 2 System utilization: (**a**) CPU Utilization, (**b**) Disk I/O Utilization.

a)		Java VM	
		Vendor A	Vendor B
D	Derby	Config. A 54.01%	Config. C 53.44%
B	MySQL	Config. B 54.63%	Config. D 54.18%
b)		Java VM	
		Vendor A	Vendor B
D	Derby	Config. A 66.15%	Config. C 69.99%
B	MySQL	Config. B 62.19%	Config. D 60.96%

2.2 Models

We now consider modeling techniques for MOM systems that can be used for performance prediction. We provide an overview of several modeling approaches and then focus on one of them showing how it can be applied in the context of a case study. In [9], an analytical model of the message processing time and throughput of the Web-SphereMQ JMS server is presented and validated through measurements. The message throughput in the presence of filters is studied and it is shown that the message replication grade and the number of installed filters have a significant impact on the server throughput. A more in-depth analysis of the message waiting time for the FioranoMQ JMS server is presented in [10]. The authors study the message waiting time based on an $M/G/1-\infty$ queue approximation and perform a sensitivity analysis with respect to the variability of the message replication grade.

Another method for modeling MOM systems using performance completions is presented in [12]. Performance completions provide a general mechanism for including low-level details of execution environments into abstract performance models. The authors propose a pattern-based language for configuring the type of message-based communication. Model-to-model transformations are used to integrate low-level details of the MOM system into high-level software architecture models.

In [11], an approach to predicting the performance of messaging applications based on the Java Enterprise Edition is proposed. The prediction is carried out during application design, without access to the application implementation. This is achieved by modeling the interactions among messaging components using queueing network models, calibrating the performance models with architecture attributes, and populating the model parameters using a lightweight, application-independent benchmark. The latter allows to avoid the need for pro-

totype testing in order to obtain the value of model parameters, and thus reduces the performance prediction effort.

The above techniques are all based on coarse-grained models focused on predicting the average system throughput and message waiting times. In [20], we propose a more fine-grained technique that allows to model the individual message flows and interactions in a MOM system. In the rest of this section, we briefly introduce this technique and present a case study that demonstrates its effectiveness.

Our modeling technique is based on Queueing Petri Nets (QPNs) [19] which are a combination of queueing networks and stochastic Petri nets. QPNs can be seen as an extension of stochastic Petri nets that allows *queues* to be integrated into the places of a Petri net. A place that contains an integrated queue is called a *queueing place* and is normally used to model a system resource, e. g., CPU, disk drive or network link. Tokens in the Petri net are used to model requests or transactions processed by the system. In our case, tokens represent the messages processed by the MOM server. Arriving tokens at a queueing place are first served at the queue and then they become available for firing of output transitions. When a transition fires it removes tokens from some places and creates tokens at others. Usually, tokens are moved between places representing the flow-of-control during message processing. QPNs also support so-called *subnet places* that contain nested QPNs. For further details on QPNs, the reader is referred to [19].

Case Study

To demonstrate the effectiveness of our modeling approach, we conducted a detailed case study [20] of a representative state-of-the-art messaging application – the SPECjms2007 benchmark – deployed on a leading commercial MOM platform. We briefly discuss the way the benchmark was modeled. Figure 3 shows the QPN model of the first benchmark interaction. The workflow of the interaction can be traced by following the transitions in the order of their suffixes, i. e., I1_1, I1_2, I1_3, etc. For each destination (queue or topic), a subnet place containing a nested QPN (e. g., DC_OrderQ) is used to model the MOM server hosting the destination. Similarly, the clients exchanging messages through the MOM infrastructure (HQ, SMs, DCs and SPs) are modeled using subnet places. Each subnet place is bound to a nested QPN that may contain multiple queueing places representing logical system resources available to the respective client or server components, e. g., CPUs, disk subsystems and network links. The respective physical system resources are modeled using the queues inside the queueing places. Multiple queueing places can be mapped to the same physical queue. For example, if all destinations are deployed on a single MOM server, their corresponding queueing places are mapped to a set of central queues representing the physical resources of the
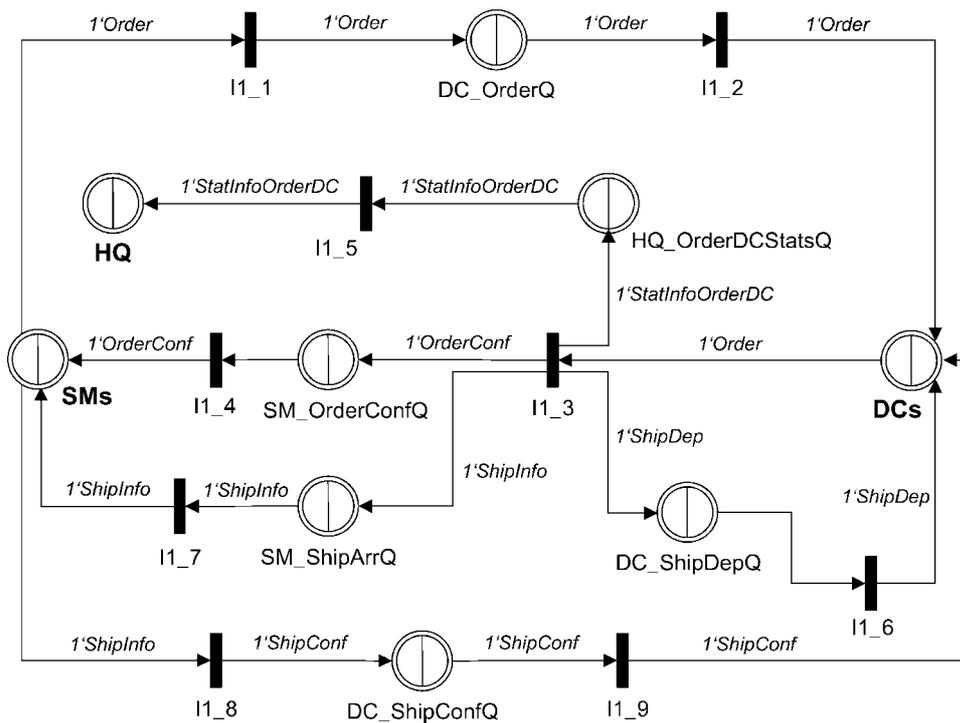
Figure 3 Model of Interaction 1 of SPECjms2007.

MOM server. The hierarchical structure of the model not only makes it easy to understand and visualize, but most importantly, it provides flexibility in mapping logical resources to physical resources and thus makes it easy to customize the model to a specific deployment of the benchmark. The remaining six interactions were modeled using the same approach.

We validated our model by conducting an experimental analysis of SPECjms2007 in the environment depicted in Fig. 1. The first step was to customize the model to our deployment environment. The subnet place corresponding to each destination was mapped to a nested QPN containing three queueing places connected in tandem. The latter represent the network link of the MOM server, the MOM server CPUs and the MOM server I/O subsystem, respectively. Given that all destinations are deployed on a single physical server, the three queueing places for each destination were mapped to three central queues representing the respective physical resources of

the JMS server. As to the subnet places corresponding to the client locations (SMs, HQ, DCs and SPs), they were each mapped to a nested QPN containing a single queueing place whose queue represents the CPU of the respective client machine. In our setup, all instances of a given location type were deployed on the same client machine and therefore they were all mapped to the same physical queue.

We employed the QPME tool (Queueing Petrinet Modeling Environment) to build and analyze the model [21]. QPME greatly simplified the task by providing a user-friendly graphical user interface for constructing QPN models and an optimized simulation engine SimQPN for steady-state analysis.

To evaluate the accuracy of the model, we studied a number of workload scenarios under different interaction mixes and workload intensities. Figure 4 shows the predicted and measured CPU utilization of the MOM server when varying the *BASE* between 100 and 700. In
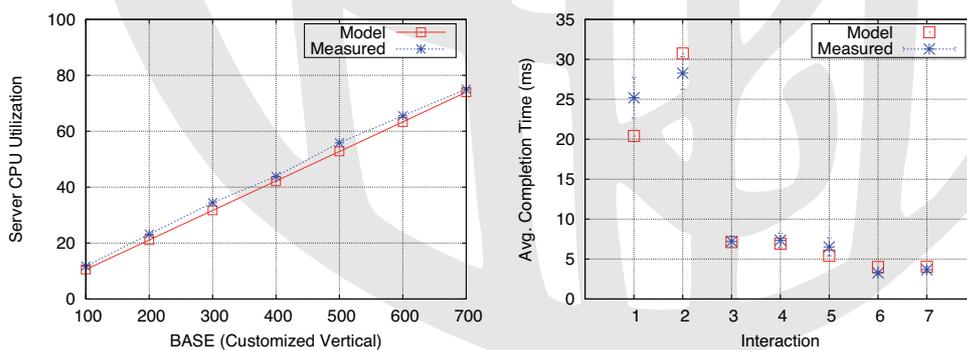

Figure 4 Model predictions compared to measurements.

addition, the average interaction completion times are shown. For completion times, we show both the predicted and measured mean values where for the latter we provide a 95% confidence interval from 5 repetitions of each experiment. The modeling error does not exceed 20% with exception of the cases where the interaction completion times are below 3 ms, e. g., for Interactions 6 and 7. In such cases, a small absolute difference of say 1 ms between the measured and predicted values (e. g., due to some synchronization aspects not captured by the model) appears high when considered as a percentage of the respective mean value given that the latter is very low. However, when considered as an absolute value, the error is still quite small. For further details on the case study including some more detailed results, we refer the reader to [20]. The model proved to be quite accurate in predicting the system performance, especially considering the size and complexity of the system that was modeled. The proposed modeling technique can be used as a powerful tool to detect system bottlenecks and ensure that systems are designed and sized to meet their performance requirements.

3 DEBS

A generic large-scale distributed event-based system (DEBS) is normally composed of a set of nodes deployed in a distributed environment and exchanging information through a set of communication networks as illustrated in Fig. 5. Clients of the system are either publishers or subscribers depending on whether they act as producers or consumers of information. Publish-

ers publish information in the form of *events* which are commonly structured as a set of attribute-value pairs. Subscribers express their interest in specific events through *subscriptions*. The main task of the system is to deliver published events to all subscribers that have issued matching subscriptions. The typical architecture of a DEBS can be decomposed into four logical layers: network layer, overlay layer, event routing layer and event matching layer. A detailed overview of the four layers as well as the techniques used to implement them can be found in [13].

3.1 Benchmarks

To the best of our knowledge, currently no benchmarks exist specifically targeted at DEBS. An overview of relevant QoS metrics in the context of distributed publish/subscribe systems can be found in [14]. Some general guidelines for designing a benchmark suite for distributed publish/subscribe systems are presented in [17], however, no specific implementation or measurement results are provided. The SPECjms2007 and jms2009-PS workloads [7] we discussed earlier can be used as a starting point for building a benchmark targeted at DEBS.

3.2 Models

Modeling DEBS is challenging because of the decoupling of communicating parties, on the one hand, and the dynamic changes in the system structure and behavior, on the other hand. When a request is sent in a traditional request/reply-based distributed system, it is sent directly to a given destination which makes it easy to
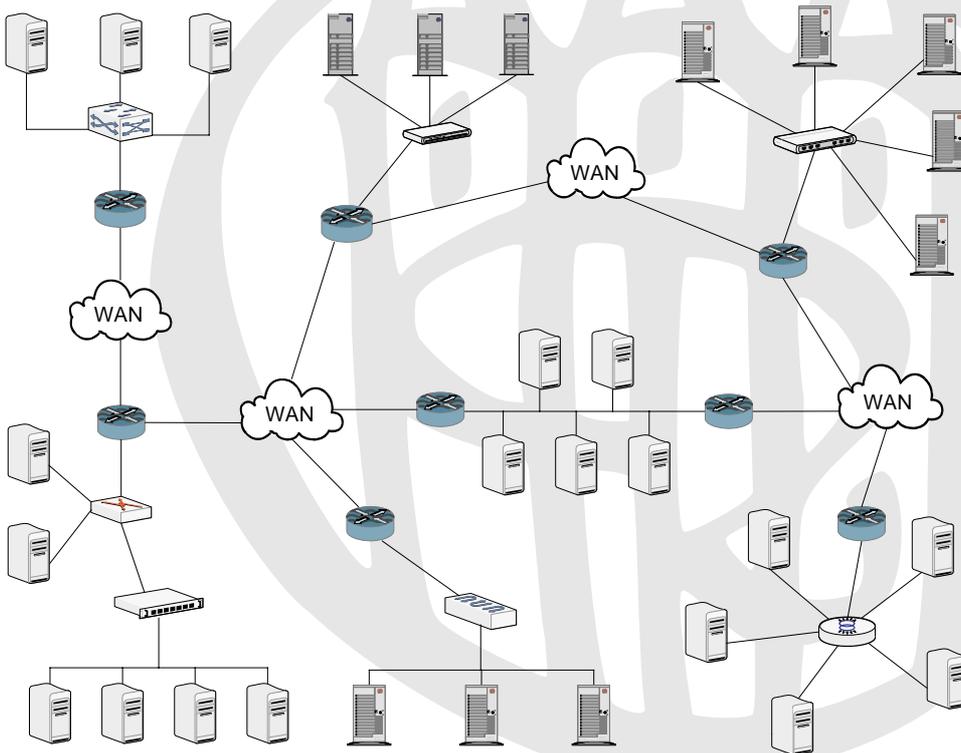

Figure 5 A distributed event-based system (DEBS).

identify the system components and resources involved in its processing. In contrast to this, when an event is published in a DEBS, it is not addressed to a particular destination, but rather routed along all paths that lead to subscribers with matching subscriptions. It is hard to know in advance which system nodes will be involved in delivering the event. There are many alternative event routing algorithms and they all have different implementation variants leading to different routing behavior. Moreover, depending on the subscriptions that exist, individual events published by a given publisher might be routed along completely different paths visiting different sets of system nodes. Another difficulty stems from the fact that every time a new subscription is created or an existing one is modified, or when nodes join or leave the system, this might lead to significant changes in the workload behavior. Thus, the dynamics of DEBS necessitate that workload characterization is done on a regular basis in order to reflect the changes in the system configuration and workload.

In [16], we proposed a novel approach to workload characterization and performance modeling of DEBS aiming to address the above challenges. We developed a workload model based on monitoring data that captures the system routing behavior and resource consumption at a level that allows us to use this information as input to performance models. The workload model we proposed does not make any assumptions about the algorithms used at the event routing and overlay network layers of the system. Using the workload model and applying operational analysis techniques we were able to characterize the message traffic and determine the utilization of system components. This in turn enabled us to derive an approximation of the mean event delivery latency. For more accurate performance prediction, we proposed detailed performance models based on Queueing Petri Nets (QPNs).

Our approach was the first to provide a comprehensive methodology for workload characterization and performance modeling of DEBS that is applicable to a wide range of systems. The methodology helps to identify and eliminate bottlenecks and ensure that systems are designed and sized to meet their QoS requirements.

In [18], an analytical model of publish/subscribe systems that use hierarchical identity-based routing is presented. The model is based on continuous time birth-death Markov chains. Closed analytical solutions for the sizes of routing tables, for the overhead required to keep the routing tables up-to-date, and for the leasing overhead required for self-stabilization are presented. The proposed modeling approach, however, does not provide means to predict the event delivery latency and it suffers from a number of restrictive assumptions. Many of these assumptions are relaxed in [22] where a generalization of the model is proposed, however, the generalized model is still limited to systems based on hierarchical identity-based routing.

4 Conclusion

This article presented a survey of current techniques for benchmarking and performance modeling of EBS. We first considered centralized systems based on MOM. With exception of the SPECjms2007 and jms2009-PS benchmarks, current benchmarks for MOM use artificial workloads that are not representative of real-world application scenarios. SPECjms2007 provides a standard workload and metrics for MOM performance as well as a flexible and robust tool for in-depth performance evaluation of MOM servers. We reviewed several techniques for modeling MOM systems and predicting their performance under load. In the second part of the paper, we surveyed the state-of-the-art in modeling DEBS considering both analytical and simulation-based approaches. We discussed the advantages and disadvantages of different approaches and provided references for further information.

References

[1] A. Hinze, K. Sachs, and A. Buchmann. *Event-Based Applications and Enabling Technologies*. DEBS'09, Keynote, 2009.

[2] Sonic Software Corporation. *SonicMQ Test Harness*. 2005.

[3] *JBoss JMS Messaging Performance Framework*. http://www.jboss.org/community/wiki/JBossJMSNewPerformanceBenchmark, 2006.

[4] M. R. N. Mendes, P. Bizarro, and P. Marques. *A framework for performance evaluation of complex event processing systems* In: DEBS'08 Demo Session, 2008.

[5] Apache ActiveMQ. *JMeter Performance Test*. http://activemq.apache.org/jmeter-performance-tests.html, 2006.

[6] IBM Hursley. *Performance Harness for Java Message Service*. http://www.alphaworks.ibm.com/tech/perfharness, 2005.

[7] K. Sachs, S. Kounev, J. Bacon, and A. Buchmann. *Performance evaluation of message-oriented middleware using the SPECjms2007 benchmark*. In: Performance Evaluation, Volume 66, Issue 8, 2009.

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

[9] R. Henjes, M. Menth, and C. Zepfel. *Throughput Performance of Java Messaging Services Using WebsphereMQ*. In: Proc. of ICDCSW'06, 2006.

[10] M. Menth and R. Henjes. *Analysis of the Message Waiting Time for the FioranoMQ JMS Server*. In: Proc. of the 26th IEEE ICDCS, 2006.

[11] Y. Liu and I. Gorton. *Performance Prediction of J2EE Applications Using Messaging Protocols*. In: Proc. of the 8th Int'l Symposium on CBSE, LNCS 3489, Springer 2005.

[12] J. Happe, H. Friedrich, S. Becker, and R. Reussner. *A pattern-based performance completion for Message-oriented Middleware*. In: Proc. of the 7th Int'l Workshop on Software and Performance, ACM 2008.

[13] R. Baldoni and A. Virgillito. *Distributed Event Routing in Publish/Subscribe Communication Systems: a Survey*. Technical Report 15–05, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienzia", 2005.

[14] S. Behnel, L. Fiege, and G. Mühl. *On Quality-of-Service and Publish/Subscribe*. In: Proc. of DEBS'06, 2006.

[15] S. Castelli, P. Costa, and G. Picco. *Modeling the Communication Costs of Content-based Routing: The Case of Subscription Forwarding*. In: Proc. of DEBS'07, 2007.

[16] S. Kounev, K. Sachs, J. Bacon, and A. Buchmann. *A Methodology for Performance Modeling of Distributed Event-Based Systems*. In: Proc. of the 11th IEEE ISORC, 2008.

[17] A. Carzaniga and A. Wolf. *A Benchmark Suite for Distributed Publish/Subscribe Systems*. Technical Report CU-CS-927-02, Department of Computer Science, University of Colorado, Apr 2002.

[18] M. Jaeger and G. Mühl. *Stochastic Analysis and Comparison of Self-Stabilizing Routing Algorithms for Publish/Subscribe Systems*. In: Proc. of the 13th IEEE MASCOTS, 2005.

[19] S. Kounev. *Performance Modeling and Evaluation of Distributed Component-Based Systems using Queueing Petri Nets*. In: IEEE Transactions on Software Engineering 32(7):486–502, July 2006.

[20] K. Sachs, S. Kounev, and A. Buchmann. *Performance Modeling of Message-Oriented Middleware – A Case Study*. Under review, Apr 2009.

[21] S. Kounev and C. Dutz. *QPME – A Performance Modeling Tool Based on Queueing Petri Nets*. In: ACM SIGMETRICS PER 36(4):46–51, Mar 2009.

[22] G. Mühl, A. Schröter. H. Parzyjegla, S. Kounev, and J. Richling. *Stochastic Analysis of Hierarchical Publish/Subscribe Systems*. In: Proc. of Euro-Par 2009.

Dr.-Ing. Samuel Kounev is a research fellow at Karlsruhe Institute of Technology (KIT) leading the Descartes Research Group funded by the German Research Foundation (DFG). He received a M. Sc. degree in mathematics and computer science from the University of Sofia (1999) and a Ph. D. degree in computer science from Technische Universität Darmstadt (2005). His research interests include software performance engineering, performance modeling and evaluation of distributed systems, run-time performance and resource management, benchmarking and capacity planning, autonomic and energy efficient computing. He has served as release manager of SPEC's Java Subcommittee since 2003 and as chair of the SPEC JMS working group since 2005. He is a member of the IEEE Computer Society and the ACM.

Address: Institute for Program Structures and Data Organization (IPD), Karlsruhe Institute of Technology (KIT) – Universität Karlsruhe (TH), Am Fasanengarten 5, 76131 Karlsruhe, Germany, e-mail: skounev@acm.org

Dipl.-Wirtsch.-Inform. Kai Sachs is a research assistant at TU Darmstadt and member of the Databases and Distributed Systems Group. He received a joint diploma degree in business administration and computer science from TU Darmstadt. His research interests include software performance engineering, performance modeling and evaluation of event-based systems and distributed systems in general, run-time performance management, capacity planning and benchmarking. He has served as lead developer of the SPECjms2007 benchmark. He is a member of the IEEE Computer Society, the GI and the ACM.

Address: Databases and Distributed Systems Group, Department of Computer Science, TU Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany, e-mail: ksachs@dvs.tu-darmstadt.de