On the Road to Benchmarking BPMN 2.0 Workflow Engines

Marigianna Skouradaki*
Dieter H. Roller
Frank Leymann
Institute of Architecture and Application Systems
University of Stuttgart
Germany
{skouradaki, dieter.h.roller, leymann}
@iaas.uni-stuttgart.de

Vincenzo Ferme*
Cesare Pautasso
Faculty of Informatics
University of Lugano
Switzerland
firstname.lastname@usi.ch

ABSTRACT

Workflow Management Systems (WfMSs) provide platforms for delivering complex service-oriented applications that need to satisfy enterprise-grade quality of service requirements such as dependability and scalability. In this paper we focus on the case of benchmarking the performance of the core of WfMSs, Workflow Engines, that are compliant with the Business Process Model and Notation 2.0 (BPMN 2.0[1]) standard. We first explore the main challenges that need to be met when designing such a benchmark and describe the approaches we designed for tackling them in the BenchFlow project[2]. We discuss our approach to distill the essence of real-world processes to create from it processes for the benchmark, and to ensure that the benchmark finds wide applicability.

Categories and Subject Descriptors

H.4.1 [**Information Systems Applications**]: Office Automation—*Workflow management*; K.6.2 [**Management of Computing and Information Systems**]: Installation Management—*Benchmarks*; D.2 [**Software Engineering**]: Metrics—*Performance measures*

General Terms

Benchmarking, Workflow Engine Performance, BPMN 2.0

1. INTRODUCTION

Performance benchmarking is an established practice that helps to drive the continuous improvement of technology by setting a clear standard in measuring and assessing performance. Only recently there have been some proposals for benchmarks of service oriented architecture middleware tools (e.g., SOABench [2]). In this paper we focus on one specific kind of middleware: Workflow Engines (WfEs), which can be used for business process automation and service composition. For WfEs there is not yet a currently accepted benchmark, even if standard workflow modeling languages such as BPMN 2.0 are widely used in academia and industrial practice. A possible explanation on this deficiency can be given by the inherent architectural complexity of WfEs and the very large number of parameters affecting their performance.

The main challenges we identify in benchmarking a real-world WfE are: (a) Collecting real-world process models; (b) Synthesing the benchmark workload out of real-world processes; (c) Designing the benchmark environment; (d) Assessing and selecting the BPMN 2.0 engines to be tested; (e) Characterising workloads of different actors; (f) Defining expressive Key Performance Indicators (KPIs).

In this paper, we introduce the BenchFlow approach for benchmarking WfEs. Its main goal is to address the aforementioned challenges, by defining the first benchmark for WfEs. In particular, it targets engines supporting BPMN 2.0 because, as we are going to show in this paper, this standard has gained a noticeable impact on the market.

2. BENCHMARK DESIGN

Given the challenges and complexity of benchmarking WfEs, we follow an iterative project management approach to design and release the benchmark. With each iteration, we enhance both the completeness and real-world representativeness of the benchmark, while taking advantage of early results to steer the BenchFlow project direction. We are currently planning to perform three iterations during the project's lifespan.

1st Iteration: runs a performance stress test on two selected WfEs, with both micro-benchmarks, to measure specific features of WfEs, and a workload mix that reflects all elements of the BPMN 2.0 Core[3], to simulate real-world behaviours. According to Muehlen and Recker [15] supporting BPMN 2.0 Core should already have a good coverage of the process models' regular usage of BPMN 2.0 constructs.

*Corresponding authors
[1]`http://www.omg.org/spec/BPMN/2.0/`
[2]`http://www.iaas.uni-stuttgart.de/forschung/projects/benchflow.php`

ICPE'15, January 31 - February 04, 2015, Austin, Texas, USA.

ACM 978-1-4503-3248-4/15/01 ...$15.00.
http://dx.doi.org/10.1145/2668930.2695527.

[3]`http://www.omg.org/bpmn/Samples/Elements/Core_BPMN_Elements.htm`

Actors interacting with the System Under Test (SUT) are omitted in this phase. In real-life executions the WfE uses the actors' think time to spread its load, while at this case the WfE must execute all the incoming activities immediately. For this reason this stress test responds to the worst case scenario for the WfE performance. In this iteration throughput (i.e., processes executed/time unit) is used as KPI.

2nd Iteration: targets to more open source and proprietary systems, with load, soak, and spike tests as performance tests [14], as well as the performance stress test from the 1st iteration. To get a step closer to real-life conditions, simple actor workload models are added in this iteration. The workload mix is also more complex in terms of structures, parallelism, and interaction with external services (e.g., JMS, web services), and BPMN 2.0 non-core activities. More KPIs measured in order to better address the new types of tests (e.g., latency, utilization, etc.) [16].

3rd Iteration: the sample of WfEs is further extended. The workload represents more complex interactions of the actors with the SUT. The impact of monitoring to the WfE performance will be assessed, as monitoring is a very common feature for WfEs. The workload mix will be more complex in structure, parallelism, and BPMN 2.0 elements (complete BPMN 2.0 set). Fault-models are also part of this workload mix. Finally the set of measured KPIs will be completed by offering the possibility to the user to select custom KPIs.

3. ADDRESSING THE CHALLENGES

3.1 Collect Real-World Process Models

In order to come up with a benchmark that correctly reflects the usage of a WfE in the real world, we need to collect as many process models representing real-world scenarios as necessary. Because "process equals product" [11] most companies and business organisations are not willing to share their process models with academic researchers to protect their intellectual property and their competitive advantage. To encourage sharing of the models we have signed confidentiality agreements with several companies and implemented a tool for obfuscating and anonymizing process models [20].

Without requesting models with a focus on a specific modelling language we have managed to collect 8363 models within four months from: the IBM Industry Models collection[4], the BPM Academic Initiative[5], companies we contacted and research projects we are involved in. More specifically, our collection contains: 1% WS-BPEL, 4% EPC, 7% YAWL, 24% Petri Net, and 64% BPMN Models, where 2/3 are BPMN 2.0. The large number of BPMN 2.0 models found in the collection supports our choice of developing a benchmark for the most recent standard process modelling language (BPMN 2.0).

3.2 Process Synthesis

The process models we have collected reflect a wide diversity of models (complex, long running, highly parallel etc.). In order to keep the benchmark close to real world we intend to accompany the default workload set, with a workload generator. Figure 1 depicts the methodology for the generation of the workload, which uses the following four phases:

Process Fragment Discovery: Addresses the automatic discovery of the most frequently reoccurring structures in a collection of process models. As BPMN 2.0 models can be seen as an attributed directed graph this problem can be reduced to frequent pattern discovery or subgraph discovery that are specification problems of graph/subgraph isomorphism. This problem is NP-Hard [5] and thus it is imperative to define an efficient methodology for discovering the similar structures. The sub-graphs are calculated with a naive algorithm approach, and clustered according to frequency of appearance. The ones with a frequency above a threshold are included in a new repository.

Process Fragment Refinement: The extracted parts are stored in the form of sliced BPMN 2.0 code. Their refinement as "Process Fragments" (namely, process parts of relaxed completeness) [19] will make this code reusable. We are initially focusing on Schumm's definition but we will tailor it to our needs. For example in BPMN 2.0, an event gateway followed by events could be considered as a fragment even if it does not include an activity. "Process Fragments" can be used to synthesize processes that will represent the existing collection.

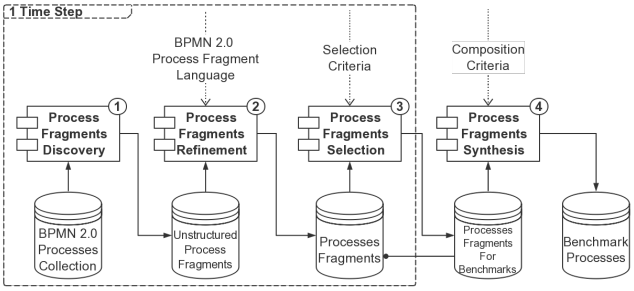

Figure 1: Workload Generation Methodology

Process Fragments Selection: All process fragments will not necessarily be of benchmark interest. This phase automatically selects fragments that satisfy benchmark related criteria that are calculated according to a set of process model metrics [13, 3]. The selected fragments are stored in a separate repository that is a subset of the initial process fragment repository.

Process Fragments Synthesis: Synthesizes the process fragments into processes according to composition criteria that are given by the user and stores them in a repository. For example, when the selection criteria ask for a process with depth $\leq$ N and M external interactions, the appropriate fragments are chosen to synthesise it. Phases 1-3 may only be executed one time, as it is not needed to extract the fragments every time.

3.3 Design the Benchmarking Environment

A carefully designed benchmarking environment and an efficient and flexible deployment mechanism are fundamental in order to guarantee the quality of the benchmark. Portability, scalability, simplicity, vendor neutrality, repeatability, and efficiency are characteristics that any reliable benchmark

[4] http://www-01.ibm.com/software/data/industry-models/
[5] http://bpmai.org/BPMAcademicInitiative/

should demonstrate [6, 9]. We setup the benchmarking environment on different physical machines on the same local network, and deploy different actors and components of the WfE on them. To obtain a "clean" measurement of the WfE performance we must separate it to the maximum possible degree from the external interferences (e.g. DBMS server and Web Services). Figure 2 gives an overview of the BenchFlow benchmark environment and how different components and main actors of the WfE are deployed on different machines.

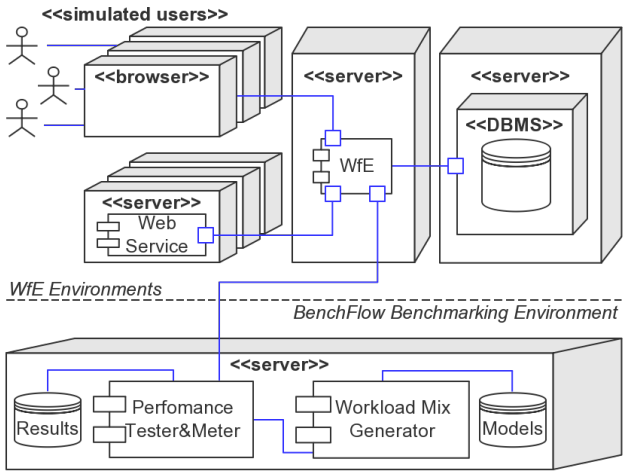

Figure 2: Benchmarking Environment Deployment

BenchFlow offers different types of performance tests (see Section 2), allowing the configuration of the workload mix (through the *Workload Mix Generator* component), the WfE under test, the characteristics of the interacting actors and the computed KPIs. Moreover it generates workload-configurable actors [Req. (1)] as the dual of benchmark processes and test configurations using a model-driven approach. This means we need to ensure a flexible deployment mechanism [Req. (2)], e.g., what is offered by tools such as Vagrant[6], a tool for managing virtual machines via a simple to use command line interface, or Docker[7], a tool for creating and working with containers to deploy complex applications. Docker fits our needs because it guarantees a sufficient level of isolation and a quick start up. It is more lightweight and requires less resources in contrast to the virtual machines approach implemented by Vagrant. Docker also allows to configure the hardware resources of different machines in a flexible way [Req. (3)], so we can use it to switch between test configurations. After the deployment, BenchFlow runs the tests using the interfaces exposed by the WfE, injecting the load as configured by the test configuration. Given that the interfaces exposed by the WfEs are heterogeneous (non-standard APIs), we map them to a common, uniform access mechanism [Req. (4)] to guarantee the best level of scalability of the benchmarking environment so that the effort to benchmark a new WfE is minimised. BenchFlow also ensures that the initial state of the different components is the same for every execution (frozen initial conditions) [Req. (5)], and verifies the environment [Req. (6)] to ensure the fulfilment of the conditions needed to execute a reliable benchmark. During or at the end of the test executions, BenchFlow collects the data to compute KPIs in a reliable way. Depending on the options of the SUT, data to compute KPIs are collected during the test executions, using tools like Faban[8] and JMeter[9], or exploiting the log generated by the WfE, which are retrieved and analysed at the end of the test execution. Moreover BenchFlow collects system performance metrics from every environment shown in Figure 2, such as CPU, memory and bandwidth usage, while minimising the invasiveness of the measurements [Req. (7)]. These metrics allow us to check for external interferences during the benchmark measurement, and guarantee the same environment conditions for each benchmarked engine.

[6] `http://www.vagrantup.com`
[7] `https://www.docker.com`

3.4 BPMN 2.0 Engines Assessment

The WfEs that participate in the benchmark need to fulfill the following requirements: a) support at least the BPMN 2.0 Core, b) be testable in order to automate the benchmark executions. Testability means that the WfE exposes APIs to interact with it at least to: b.1) deploy a process; b.2) request to start the process execution; b.3) access pending user, manual and receive tasks, intermediate catching message and signal events; b.4) access the process execution log; and c) be still in active development.

We have conducted a survey among the existing WfEs that support BPMN 2.0, gathering the following information from the discovered products' Web sites and release notes. Our search found both proprietary and open source products. We have found 19 systems in active development that support BPMN 2.0. The complete list is omitted for space reasons but available online[10].

The release date of the first version supporting BPMN 2.0 has been used to visualize the trend of the number of systems in active development that support BPMN 2.0 over time (cf. Figure 3). In addition to show the rapid adoption of the standard (released in January 2011), this trend shows that the time is ripe for developing a benchmark for it.

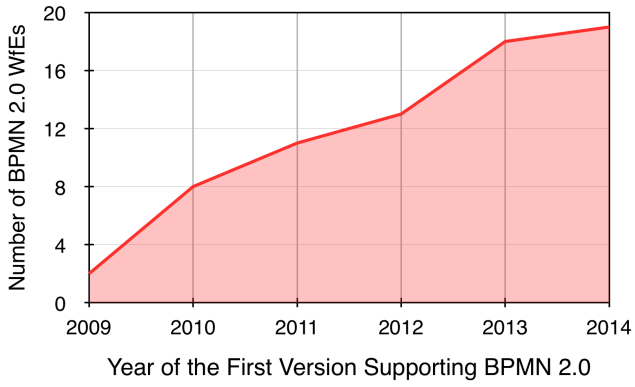

Figure 3: Trend of the Number of WfEs Supporting BPMN 2.0

The systems are written in different programming languages (PHP, Javascript, Java, etc.). This heterogeneity of programming languages makes it particularly challenging to

[8] `http://faban.org`
[9] `http://jmeter.apache.org`
[10] `https://en.wikipedia.org/wiki/List_of_BPMN_2.0_Engines`

compare their performance. We have found it surprisingly difficult to determine the level of testability of a given WfE (req. (b)) without actually installing it and looking for the necessary APIs, since these are rarely documented on the corresponding websites. For the same reason, we need to run a compliance test for the BPMN 2.0 standard as done for BPEL WfEs in [8], in order to assess the req. (a).

4. RELATED WORK

To the best of our knowledge BenchFlow is the first benchmark that specifically targets BPMN 2.0 WfEs. There is a widely recognised need for introducing such a benchmark [22, 18], which would enable the evaluation of performance of different research prototypes and commercial products in meaningful conditions.

SOABench [2] can be seen as an initial step to provide a performance assessment and comparison framework of SOA middleware systems. It features automatic generation and execution of testbeds for benchmarking BPEL WfEs. However SOABench assumes that the performance of a BPEL WfE can be reduced to its response time. OpenESB [21] and Din et al. [4] use a simple synthetic process to run the benchmark. ActiveVOS [1] and Intel Cape Clear [10] perform a load testing of a proprietary system with two real process models.

Roller [17] and FACTS [12] perform load testing using one real-world process to stress an open source and a proprietary WfE. Both of these works invoke external services through their processes. Hackman et al. [7] benchmarks BPEL WfEs using 12 kernels processes. The benchmark performs a baseline test that measures the latency and the memory utilisation of two open source WfEs.

BenchFlow is different considering: a) the number and heterogeneity of the WfEs under test, b) the growing complexity of the workload mix, and c) the type of performance tests that will observe a broader spectrum of raw performance metrics and aggregate them into meaningful KPIs.

5. CONCLUSION AND FUTURE WORK

In this paper we discussed critical aspects of the design of a benchmark for BPMN 2.0 WfEs that will help towards the comparison of the performance characteristics of different WfEs and therefore stimulate further research in this important middleware technology. We presented our initial approach to tackle some of the challenges that one meets when designing a benchmark for WfEs. Our goal is to start a discussion on our benchmarking approach within the community, interested in studying the performance of middleware for workflow and business process management and come up with a well-designed, widely accepted and usable benchmark for assessing, comparing and further improving the performance of BPMN 2.0 WfEs.

Acknowledgements

This work is funded by the Swiss National Science Foundation and the German Research Foundation with the BenchFlow (DACH Grant Nr. 200021E-145062/1) project.

6. REFERENCES

[1] Active Endpoints Inc. Assessing ActiveVOS performance, 2011. `http://www.activevos.com/content/developers/technical_notes/assessing_activevos_performance.pdf`.

[2] D. Bianculli, W. Binder, and M. L. Drago. SOABench: Performance evaluation of service-oriented middleware made easy. In *Proc. of ICSE'10 - Volume 2*, pages 301–302, 2010.

[3] J. Cardoso. Business process control-flow complexity: Metric, evaluation, and validation. *International Journal of Web Services Research*, 5(2):49–76, 2008.

[4] G. Din, K.-P. Eckert, and I. Schieferdecker. A workload model for benchmarking BPEL engines. In *Proc. of ICSTW'08*, pages 356–360, 2008.

[5] M. Dumas, L. García-Bañuelos, and R. M. Dijkman. Similarity search of business process models. *IEEE Data Eng. Bull.*, 32(3):23–28, 2009.

[6] J. Gray. *The Benchmark Handbook for Database and Transaction Systems*. Morgan Kaufmann, 2nd edition, 1992.

[7] G. Hackmann, M. Haitjema, C. Gill, and G.-C. Roman. Sliver: A BPEL workflow process execution engine for mobile devices. In *Proc. of ICSOC'06*, pages 503–508. Springer, 2006.

[8] S. Harrer, J. Lenhard, and G. Wirtz. BPEL conformance in open source engines. In *Proc. of SOCA'12*, pages 1–8, 2012.

[9] K. Huppler. The art of building a good benchmark. In *Performance Evaluation and Benchmarking*, pages 18–30. Springer, 2009.

[10] Intel and Cape Clear. BPEL scalability and performance testing. White paper, 2007.

[11] F. Leymann. Managing business processes via workflow technology. In *Proc. of VLDB 2001*, pages 729–, 2001.

[12] A. Liu, Q. Li, L. Huang, and M. Xiao. Facts: A framework for fault-tolerant composition of transactional web services. *IEEE Trans. on Services Computing*, 3(1):46–59, 2010.

[13] J. Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*. Springer, 2008.

[14] I. Molyneaux. *The Art of Application Performance Testing: Help for Programmers and Quality Assurance*. O'Reilly, 2009.

[15] M. Z. Muehlen and J. Recker. How much language is enough? theoretical and practical use of the business process modeling notation. In *Proc. of CAiSE'08*, pages 465–479, 2008.

[16] C. Röck and S. Harrer. Literature survey of performance benchmarking approaches of BPEL engines. Technical report, Otto-Friedrich University of Bamberg, 2014.

[17] D. H. Roller. *Throughput Improvements for BPEL Engines: Implementation Techniques and Measurements applied in SWoM*. PhD thesis, University of Stuttgart, 2013.

[18] N. Russell, W. M. van der Aalst, and A. Hofstede. All that glitters is not gold: Selecting the right tool for your BPM needs. *Cutter IT Journal*, 20(11):31–38, 2007.

[19] D. Schumm, D. Karastoyanova, O. Kopp, F. Leymann, M. Sonntag, and S. Strauch. Process fragment libraries for easier and faster development of process-based applications. *CSSI*, 2(1):39–55, 2011.

[20] M. Skouradaki, D. Roller, C. Pautasso, and F. Leymann. BPELanon: Anonymizing BPEL processes. In *Proc. of ZEUS'14*, pages 9–15, 2014.

[21] Sun Microsystems. Benchmarking BPEL service engine, 2007. `http://wiki.open-esb.java.net/Wiki.jsp?page=BpelPerformance.html`.

[22] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann. Monitoring and analyzing influential factors of business process performance. In *Proc. of EDOC'09*, pages 141–150, 2009.