Towards Workflow Benchmarking: Open Research Challenges

Cesare Pautasso
Vincenzo Ferme

Faculty of Informatics
University of Lugano (USI)
via Buffi 13
CH-6900 Lugano, Switzerland
c.pautasso@ieee.org
vincenzo.ferme@usi.ch

Dieter Roller
Frank Leymann
Marigianna Skouradaki

Institute of Architecture of Application Systems
University of Stuttgart
Universitätsstr. 38
70569 Stuttgart, Germany
Dieter.H.Roller@iaas.uni-stuttgart.de
leymann@iaas.uni-stuttgart.de
marigianna.skouradaki@iaas.uni-stuttgart.de

Abstract: Workflow Management Systems (WfMS) provide platforms for delivering complex service-oriented applications that need to satisfy enterprise-grade quality of service requirements such as dependability and scalability. The performance of these applications largely depends on the performance of the WfMS supporting them. Comparing the performance of different WfMSs and optimizing their configuration requires that appropriate benchmarks are made available. In this position paper we make the case for benchmarking the performance of WfMSs that are compliant with the Business Process Model and Notation 2.0 standard and explore most of the challenges that one must tackle when constructing such benchmarks.

1 Introduction

Since the introduction of Web services [Pas05], workflow-based applications [LR97] have become the state-of-the-art programming model for integrated applications and information systems. They support the implementation of the two-level programming paradigm [WWC92], where the programming in the large is carried out using process models and the programming in the small is implemented via appropriate Web Services [ACKM04, GGKS02, WCL+05]. This approach has entered all business areas [KKL06, ZTP03] rang-

ing from simple forms-oriented applications, such as opening a new account in a bank or the approval of a travel plan in a software company, to complex applications, such as the planning for the construction of a car in the design department of a car manufacturer.

The performance of workflow-based applications depends basically on two factors: the performance of the workflow management middleware and the performance of the application components [LBK03]. The application components are either produced by the company running the application or have been provided by some software vendor. These are typically delivered as a service. We assume that these components have been designed and implemented using appropriate best practices and have been configured and tuned for optimal performance. It can be further assumed that the application components and the Workflow Management Systems (WfMS) are clearly separated; a situation that is normal if the WfMS is provided by some software vendor and/or implemented using the BPMN 2.0 standard [JE11]. In this case, the achievable overall performance of the application depends solely on the performance of the WfMS.

We find ourselves now at the same situation as database management systems in 1985, when Jim Gray published the first benchmark proposal [BBC+85]: agreement had been reached on an appropriate standard and the first set of database engines were delivered as commercials systems. Proposing a comparison benchmark posed an important challenge that sparked significant research activities in the field of databases resulting in an unexpected dramatic improvement of database technology; the performance of database engines measured through the TPC benchmark improved dramatically from 54 transactions/minute in 1992 to more than 10 million transactions/minute in 2010.

In the context of WfMSs, we have a similar situation: agreement has been reached on BPMN 2.0 as the standard for Business Process Management [Ley11] and more and more commercial and research WfMSs based on BPMN 2.0 are entering the market[1].

It is the right time to develop a benchmark for BPMN 2.0 workflow engines that helps to give an objective and quantitative comparison and as a result drives forward workflow middleware technology[2].

WfMSs belong to a different class of software systems [Wes07] as opposed to database management systems or programming language compilers, for which there are established benchmarks (e.g., TPC-C [Tra97], SPEC CPU [Sta11]) and well known performance analysis and optimization techniques [Jai91]. In contrast to databases, which passively react to a workload of queries [Gra92], WfMSs actively drive the orchestration of distributed services which lie outside of their control [LRS02]. As opposed to programming languages, which can be optimized for local execution, workflow languages make use of composition constructs which shift the focus of the optimization from local computations to distributed

[1] `https://en.wikipedia.org/wiki/List_of_BPMN_2.0_Engines`

[2] Business Process Management cannot be fully reduced to BPMN 2.0, as many other approaches (e.g., WS-BPEL, Case Management, Structured vs. Unstructured, Declarative/Rule-based vs. Imperative process modeling) have been proposed and are being pursued. This heterogeneity has been so far detrimental to the development of a comprehensive benchmark to compare systems that are very diverse in terms of the features they offer, the different types of middleware technology they leverage underneath and the meta-modeling paradigm they use to describe business processes. Thus, in the rest of this paper we assume a level playing field whereby the engines to be benchmarked comply with one standard.

interactions.

The objective of this position paper is to define the challenges involved in the construction of a solid benchmark for BPMN 2.0 based WfMSs that helps 1) analyze the performance behavior of WfMSs, 2) give a fair comparison to different systems and therefore, to 3) contribute to drive forward the progress of workflow technology.

The rest of this position paper is organized as follows: Section 2 presents work that is related to benchmarking WfMS and examines its relevance with respect to our goals. Section 3 discusses the main factors that contribute to determine the performance of a WfMS. Sections 4 and section 5 explore the logistic and technical challenges that designers of WfMS benchmarks must overcome. Section 6 discusses the relationship between the challenges and the performance factors, while Section 7 concludes this position paper.

2 Related Work

When it comes to evaluating the performance of complex systems such as modern service oriented architectures [GGKS02], there has been a lot of work focusing on each layer of the architecture [HZ06], going from the storage layer [Gra92, TZJW08, Cha95] all the way up through the middleware [BCM+05, SKBB09, GSC+04, PP08] into the Cloud [BKKL09]. In this section we present a brief summary of relevant benchmarking literature, including references to the most important surveys [WFP07].

Concerning WfMS, relatively less work can be found specifically targeting the topic of benchmarking their performance (e.g., [GMW00, BBD10a, DPZ11]). One of the first benchmarks that has been published was LabFlow-1 [BSR96]. The goal was to study the impact of the database on the workflow performance; in fact, the authors saw it more as a database benchmark rather than a workflow benchmark, comparing the overhead different database management systems used by a workflow engine. The benchmark was motivated by the statements made in [GHS95] that commercial WfMSs could not support applications with high-throughput workflows and would not meet the needs of the associated genome research center for high-performance workflows. Unfortunately, the structure of the workflows was not given, so that no judgment can be made how the benchmark would perform on a modern, state-of-the-art WfMS.

The next benchmark [GMW00], conducted in 2000 by the database group of Gerhard Weikum at the University of Saarland, was comparing the performance of a commercial WfMS with the one that has been developed by members of the group. The base was a rather simple e-commerce workflow that was described using state charts. The actual benchmark measured the throughput of each of the systems as well as the impact of the database work that was forwarded to a dedicated server. The maximum throughput of the systems was measured at 400 processes/hour running on a SUN Sparc 10. The results may have been good for the time of the benchmark; however, it can be assumed, that a state-of-the-art WfMS would significantly outperform such benchmark (in particular given today's hardware capabilities).

Hence, the need to introduce a benchmark that addresses the state-of-art features is widely

recognized [KKL06, WLR+09, RvdAH07, LMJ10]. As one could speculate the vendors of commercial systems are running internal benchmarks (e.g. [IBM11], [Act11], [IC07]) to come up with performance figures that they publish in their product documentation or provide to prospective customers upon special requests. However, there is still no commonly accepted way to compare the performance of different engines.

Towards this direction SPEC introduced in 2010 a subcommitee on SOA [Sta10] middleware benchmarking. The scope of the working group is rather broad, as it covers business process choreography among many other SOA middleware technologies.

More recently, SOABench [BBD10a] is a framework for the automatic generation, execution and analysis of test beds for evaluating the performance of service-oriented middleware. It has been used in [BBD10b] to compare the performance of several WS-BPEL engines (ActiveVOS, jBPM, and Apache ODE). The goal of the benchmark was to test the efficiency of the individual structural activities, such as `<sequence>`, `<flow>` with and without links, and `<while>`. Each activity type was used in a process model with five invoke activities, whose implementation was a simple servlet. The performance was compared using a different number of clients and different think times between subsequent requests. Unlike the work in [GMW00], the measurements focused on response time and identified several scalability limitations of the systems being tested.

A black-box approach to workflow benchmarking has been published in [DPZ11]. The goal of the authors is to compare the performance of five unnamed workflow engines for the XPDL language under various workload conditions. The main metric chosen is the completion time and the proposed reference processes only involve local computations (e.g., incrementing loop counters and generating random numbers). The outcome is that commercial engines outperform the freely available ones under heavy load conditions.

OpenESB [Sun07] and Din et al. [DES08] use a simple synthetic process in their benchmarks. The first focuses on load testing while the second one on stresses the response time behaviour. Roller [Rol13] and FACTS [LLHX10] perform load testing using one real-world process to stress an open source and a proprietary engine. Both of these works invoke external services through their processes. Silver [HHGR06] benchmarks two BPEL engines using 12 kernel processes. It performs a baseline test that measures the latency and the memory utilisation.

The challenges that are analysed in the sections 4 and 5, have as a goal to extend the aforementioned related work, with respect to the following key points: a) a workload mix of diverse complexity, and b) performance tests of different types that will consider a larger set of raw performance metrics and aggregate them into meaningful key performance indicators.

3 Performance Factors

The contents of a benchmark and the associated challenges for constructing such a benchmark can only be appreciated if one understands how the various aspects of a WfMS will impact the performance with which a WFMS can execute business processes (Figure 1).

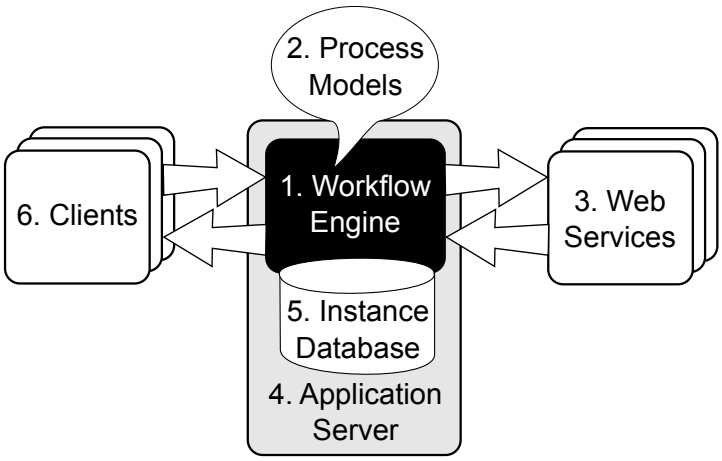

Figure 1: Factors affecting the performance of a WfMS

Six areas can be distinguished that contribute to the performance of a WFMS: (1) The architecture of the workflow engine, (2) The structure and complexity of the business processes that are being carried out, such as the level of parallelism or the handling of data, (3) The interactions that the business process carries out such as responding to a request from a client or invoking a Web Service, (4) The exploitation of the underlying middleware, such as transaction handling or message processing, (5) The management of data in a persistent store for the reliable execution of long-running processes, and (6) the load that the WfMS must sustain.

3.1 Workflow Engine Architecture

The internal architecture of the workflow engine is a very important factor contributing to the performance of the overall workflow-based application. Typically BPMN 2.0 engines will transform the process model into an internal representation that can be executed efficiently. In some cases, they may introduce several pools of worker threads to pipeline the concurrent execution of multiple, independent process instances and rely on the underlying application server middleware services to efficiently manage synchronous and asynchronous interactions. To achieve scalability, some engines have been designed to replicate key components to run across multiple processor cores, or even in a distributed environment such as a cluster of computers or a Cloud [PHA07]. There are also many possibilities concerning the performance optimizations that have been applied to modern workflow engines. These involve aspects such as:

• transaction management: where ACID properties need to be guaranteed both towards the external services participating in the workflow as well as within the engine, which is responsible for the consistent and dependable management of the state of each process

instance. Transaction boundaries can be adjusted to minimize conflicts, especially concerning short-lived micro-flows [HZ06].

• message processing: buffering, filtering, normalization, sorting, aggregation, and correlation of messages that are exchanged with the environment through the underlying middleware. Zero-copy or pass-by-reference techniques can be introduced as long as they do not violate the semantics of the process modeling language.

• failure handling and recovery: it is a common trade off to provide support for the persistent and reliable storage of the state of process instances by giving up some performance. This is particularly important for long running processes, which are likely to be affected by failures affecting the workflow execution engine itself. Write-through caching, persistent message queues and optimized logging representations can potentially make a big impact on the performance [LR98].

• monitoring: dynamically generating up-to-date reports on the internal activities of the system may require to combine OLAP characteristics within an architecture whose primary goal is to drive forward the execution of the active workflows, which is a form of OLTP [BGNS10]. Not only it is important to observe the impact of monitoring on the performance of the raw workflow engine, but also monitoring itself can be a source of specific benchmarking scenarios and challenges.

Having a benchmark would make it possible to fix the boundary conditions around the workflow engine (i.e., process models, interactions, workloads) so that it would be possible to measure and quantitatively compare the effect of critical design decisions on the engine's architecture.

3.2 Process Model Complexity

The complexity of a process model plays a significant role in the performance that can be achieved. It poses a significant challenge on the design and implementation to make sure that one does not pay for constructs if they are not defined in a business process. The complexity of a process model has many facets [Car07]; we illustrate here the most prominent ones which mostly affect the runtime performance of the process.

One important facet is the structure of the process model in terms of parallelism. Parallel processing within a business process can improve process execution time, in particular if time dependent processing, such as `timer` events, is available or other Web Services are invoked. However, extra execution resources are needed by the WfMS to cope with the parallel processing of each parallel path, including the safe handling of variables that are concurrently referenced in multiple paths.

Processing of data in transition conditions and assign activities is another important part of executing a business process. Depending on the size of the data stored within the involved variables and the level of sophistication of the XPath expressions, handling of data may require significant execution resources by the WfMS.

Compensation processing mandates that data and execution histories are kept so that pro-

cessing can be undone or alternate processing can be carried out. Storing persistent execution logs requires sufficient storage resources and also imposes an overhead during the execution of a transactional business process.

3.3 Interactions

The interactions that a business process carries out fall in two categories: the processing of client requests and the invocation of external Web Services. These are carried out using one of the following three interaction patterns: (1) synchronous request-reply, (2) asynchronous message-exchange, and (3) fire-and-forget. All patterns are defined via appropriate BPMN constructs (Figure 2) such as service task for (1), throw-catch of a message event or send-receive tasks for (2) and throw of a message event or send task for (3).

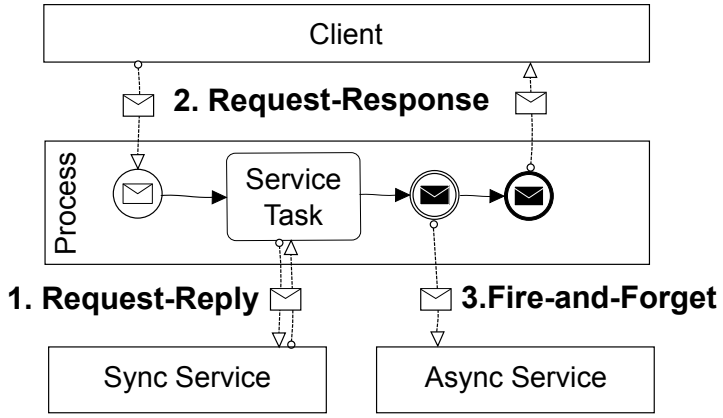

Figure 2: Interactions Handled by a WfMS

3.3.1 Request-Reply

The requestor starts the request-reply interaction pattern by calling the Web Service via the appropriate interface and then waits for the invoked Web Service to return the defined output value. When a client issues such a request, the WfMS loads the associated process model, creates a process instance, and has the appropriate receive handle the input value supplied in the request. Processing of the process instance continues by running the activities in the prescribed order, including possibly some interactions with other Web Services. The business process issues such requests, when a service task is defined. The WfMS in this case calls the defined Web Service and then waits until the response comes back. Finally, when an end state is reached, the associated variable is returned to the client. Even if

this pattern is designed for very short interaction in the range of seconds, the WfMS cannot make any assumptions about the response time of a called Web Service (when calling out) nor about the time does it take to execute a process instance (which determines for how long time the client remains connected to the engine waiting for a reply).

3.3.2 Request-Response

A requestor starts this pattern by calling the Web Service asynchronously; it is not waiting for any response, the response comes back to the requestor by having the invoked Web Service calling the requestor back at a specified end point reference. The requestor message contains correlation information that is returned by the Web Service along with its call back and that allows the requestor to correlate the response to the original request.

If the requestor is a client, the WfMS in the receive activity or in the catching message event extracts the correlation information and makes the extracted information part of the value returned in the response message. If a business process is the requestor, the Web Service is invoked via an appropriate send task or throwing a message event with correlation information being part of the input value.

Processing of the business process continues until a receive activity or an itermediate catching message event is encountered that has been defined with the same partner link. The WfMS now stops processing until a message is received for the receive activity or the catching message event. The correlation information in the message is used to locate the appropriate process instance (the one that issued the request).

As in the case of the request-reply, the WfMS can make no assumption about the time it takes for the called Web Service to respond. It must be able to process instantaneous responses (even within milliseconds) as well as responses that come after a very long time (e.g., more than a month).

3.3.3 Fire-and-Forget

This pattern is a simple asynchronous call, either issued by a client or by the business process. There is nothing special about this type of request, except that exception situations need very special treatments. For example, a business process invokes a Web Service which completes successfully, however the process aborted and may be automatically restarted since the processing of a WfMS is typically carried out within transactional brackets. From a performance perspective, these interactions are the ones which pose the least burden on the system, since they do not block and do not usually include correlation metadata.

3.4 Application Server Exploitation

WfMSs typically rely on functions that application servers provide, such as application scheduling, or timer services. The following functions are the most prominent ones from

a performance perspective.

Support of transactions All processing of a WfMS is carried out as transactions, so that a WfMS automatically provides forward recoverability; that means when the system fails, all changes applied to the business process are undone and the system is restarted with the original request. Note that backward recovery for business processes is provided via the compensation processing.

Queues Most WfMSs use message queues for interaction between the different components. These queues might become performance bottlenecks with concurent components subjected to heavy load.

Timer Services The timer event of the BPMN 2.0 specification as well internal functions of the WfMS exploit the timer services for starting appropriate actions at the right time.

Application management The various components of the WfMS are managed by the application server. This includes managing the execution lifecycle of the components that make up the WfMS, including dynamic loading and unloading of components.

3.5 Database Usage

Most business processes are long-running that means they cannot be carried out in a single execution step; those that do are typically called micro flows. This mandates that the WfMS stores the state of those process instances in some persistent store. The amount of data that must be stored thus becomes highly dependent on the execution time of process instances. The database is not only used by the WfMS internally but also to answer process queries, a facility that most WfMSs provide. In addition, most WfMSs provide audit trailing, a facility that writes an entry into a log table for each change in the state of a process instance. If audit trailing is activated, additional history data is generated and needs to be indexed and stored persistently. Extra storage is required if process instances must be, e.g., for legal reasons, kept for many years. Even with rapidly decreasing storage costs, the amount of storage required by different workflow engines is an important cost factor and may significantly change depending on the offered level of persistence.

3.6 Load/Request Management

A WfMS must sustain a plethora of different request scenarios [DES08], such as a large number of clients issuing moderately low-frequency requests, a small number of clients rapidly firing off many requests, or number of requests peaking at certain times during the day. Furthermore, users are expecting that WfMSs to some extent scale with the hardware that is available. Clustered or distributed workflow engine architectures which may be deployed on multicore servers on in the Cloud are starting to appear [ALMS09, LMJ10]. Such engine architectures may trade-off response time against throughput, making the

choice of workload very critical to give a fair assessment of their performance. Furthermore the chosen workload may be domain-specific [YÖK03], thus depend on the type of processes that are to be executed as part of the benchmark.

4 Logistic Challenges

This section introduces the set of challenges that one will find when trying to obtain information that correctly reflects the usage of WfMSs in the field.

4.1 Collecting Real World Scenarios

The notion of workflow-based applications [LR97] has entered all domains of applications, ranging from simple transaction-oriented processing in a bank, straight-through processing in financial institutions, the Enterprise Application Integration (EAI) scenarios of a production company, to the support of engineers in the design department of a car manufacturer. It is therefore necessary to collect as many real world scenarios as necessary to have a fair representation of the applications, which use WfMS as their base. Only then it is possible to come up with a benchmark that correctly reflects the usage of the WfMSs in the real world. The challenge here is to only collect as many scenarios as necessary but not more. It is very likely that a single benchmark can hardly reflect the diversity of applications or the different exploitation of workflow technology in the different industries, so one may end up with several benchmarks reflecting the level of sophistication of the WfMS and orthogonally to it several benchmarks for different industries. In this case, it may be required to collect more real world scenarios for the construction of the different benchmarks. It is understood that getting real world scenarios is also a challenge, since many companies do not want to disclose their business processes for competitive reasons.

4.2 Synthesizing Benchmark Flows

The underlying process models reflect the diversity of the applications, ranging from simple sequential processes with a few activities to complex, long running, highly parallel processes, from the invocation of simple synchronous Web Services to highly complex, context-based message exchange scenarios in order fulfillment processes. To keep the benchmark manageable, it is virtually impossible to construct a benchmark by just adding processes corresponding to each particular scenario for each usage segment. Thus one needs to extract the essence of each of the scenarios and construct a set of synthetic process models plus appropriate Web Services. The challenge is therefore to construct a minimal number of process models that correctly reflects the actual usage of the different process model constructs that the process model language offers. In other words, the set of process models making up the benchmark needs to reflect the usage pattern of the various

BPMN 2.0 constructs in real-world scenarios. We envisage a modular set of benchmarks including the mostly used BPMN 2.0 features in the core and the less frequently used features as optional extensions. For this purpose we intend to implement a static and dynamic analysis on the selected collection of process models. These analyses will export micro characteristics of process models such as which language elements are mostly used and to which extent, and also macro characteristics, which are namely reoccurring structures (e.g., fragments and patterns) in the collection of process models. This way we can automate the synthesis of representative processes.

4.3 General vs. Domain Specific Benchmark Flows

In a second step, the synthesized process models must be aggregated in such a way that they correctly reflect the appropriate usage patterns of the different scenarios; that means the execution of the different process models with their different process activities should reflect the sum of all scenarios, if possible. If that does not prove to be feasible, it is suggested to develop a set of benchmarks that reflect certain industries, such as banking or public sector. The cross-industry, function exploitation oriented approach has been also followed for the database benchmarks through the introduction of special benchmarks [PF00], such as the TPC-D [Tra95] benchmark for data warehouses, or the TPC-H [Tra] for decision support systems. Given the large number of application domains in which workflow technology has been successfully applied, it is challenging to select a suitable representative subset of domains from which to start from collecting and synthesizing domain specific benchmark process models.

4.4 Benchmark Number and Key Performance Indicators

The result of the benchmark should be preferably a single number (or as few values as possible). Only a single number allows the easy comparison of different WfMSs. Special care must be taken that the benchmark number is fair, that means that is not favoring particular architectures or implementations. Given the very large number of metrics that can be used in observing the performance of a WfMS [LBK03, WLR+09] (just to name a few examples: throughput, response time, resource and power consumption, network traffic consumed, peak number of concurrently active process instances, total number of started vs. completed process instances, SLA violations, failure rate, required memory/disk storage, recovery time) it will particularly challenging to select and aggregate them into a single meaningful number. Additionally, it will be necessary to choose between a black-box approach, relying on metrics which can be obtained non-intrusively, or a white-box approach which may require to instrument the workflow engine to observe its internal behavior in detail.

It is suggested that one follows a similar approach to what is currently done in database benchmarking where the obtained performance is provided as an absolute number (trans-

actions/minute) as well as cost-rated number (transactions/minute/euro). However, it is also important to investigate if response time would make a useful comparison metrics (or whether there is a strong correlation between response time and throughput).

5 Technical Challenges

This section presents a set of technical challenges that one will encounter during the design of a fair benchmark that takes into account specific features and characteristics of WFMS.

5.1 Application Implementation Impact Elimination

The database benchmarks are conceptually simple: a test client fires off a number of requests against the database management system. The test completes when all pre-defined request have been issued. This is quite different for benchmarking a WfMS where not only the workflow engine is involved but also the invoked Web Services contribute to the end-to-end performance; both require appropriate hardware resources (CPU, I/O, and Network). The challenge here is the elimination of these non-WFMS resources from the benchmark results so that only the WfMS proper is benchmarked. The typical approach of eliminating the application impact by short-cutting the invocation does not work for at least two reasons: (1) one has no access to the internal code of the engine for engineering the short cut, (2) the interaction between a process and its invoked Web Services is normally carried using conversation-based message exchange protocols.

5.2 Prevent System Overloading

The WfMS exposes its activities that receive messages from the outside as Web Services. Thus the interaction between the invoked Web Services and the process is via set of Web Service calls either invoked by the WfMS for the Web Services implementing the invoke activities in the process or by the invoked Web Service calling the Web Services that implement the receive activities of the process. The same is true in general for the clients requesting the execution of a process; the process either sends an asynchronous response back to the caller, or returns the result via a call back to the calling Web Service. Consequently a simple test client like the one that is constructed for database benchmarking cannot be used. The test client in database benchmarks just fires off requests and waits until the database management system comes back with an answer before submitting the next request. This notion of having the client carrying out synchronous calls will not work for driving the execution of processes in a WfMS. Thus the client can only fire off requests using some delay between the individual requests so that the WfMS is capable of handling all requests without overloading, thrashing or keeping requests buffered somewhere and eventually being throttled by the underlying messaging infrastructure. The challenge

would be to automatically adjust the delay rate so that the WfMS can cope with the requests.

5.3 Benchmarking Long Running Processes

The lifetime of processes ranges from milliseconds for a process that invokes another Web service to obtain, for example, the price of a particular book in a book store service to seconds for the handling of a sales transaction in straight-thru-processing, to weeks for an order process that handles the ordering of goods from a supplier, or even years for a process that controls the design process at a car manufacturer. The characteristics of those long running processes are twofold: (1) the instance database gets large and (2) neither the WfMS nor the database management system can take advantage of caching. The challenge is to find a method/setup that allows the benchmarking of those long-running processes without having to wait for years in order for the benchmark to complete.

5.4 System Internal Load Optimization

Some WfMSs exploit internal load optimization techniques to cope with the situation that the request load quite often varies with the time of the day. Thus they shift work to the time of the day of when the request load is lower. For example, IBM MQSeries Workflow postpones the resource-intensive deletion of process instances to lower-activity times. Likewise, Microsoft Windows Workflow Foundation can be configured not to always immediately move completed process instances to a specific database partition. The challenge is to accommodate this kind of adaptive/optimized behavior in the measurements for the throughput.

5.5 Correlation-based Message Exchange

Interactions between the invoked Web Services and the process are carried out as set of message exchanges. The messages are tagged with user-defined correlation information that helps the receiving WfMS and the invoked Web Services to find the appropriate process instance or other state information, in case the Web Service is not implemented as a process. Since the associated correlation information must be unique during the life time of a particular process instance, the challenge is to design and develop a test client in such a way that unique correlation tokens are generated when a request is sent to the WfMS for starting a process instance.

5.6 Performance Impact of Workflow Language Features

Workflow languages in general and BPMN 2.0 in particular provide a rich set of constructs to model iteration, concurrency and parallelism, exception handling and failure recovery, etc. During the design of the benchmark it will be important to consider whether all of the possible features of the workflow language will have to be covered by the test workloads, or only the ones that are actually and most frequently used in practice [MR08].

The challenge is to accommodate a variable degree of support for advanced language features by different engines. If the benchmark will require engines that provide full support for all language features, this may excessively restrict the number of engines that can be tested. This will enable the benchmark to be used as a tool to study the impact of given language features on the performance of a WfMS implementing (or not implementing) them. The hypothesis is that some engines may avoid implementing some features for performance reasons.

5.7 Reliability, Recovery and Robustness

An important feature of workflow engines concerns the guarantee of persistent execution for the process instances. If a failure occurs the state of the process instances is reloaded from persistent storage so that their execution can recover. The challenge is to define methods for controlled failure injection experiments in order to measure the failure rate of different systems as it is observed by the clients of the workflow engine. It should be noted that failure cannot be achieved through code modifications, but could be introduced by corrupting the messages as they reach the engine. At a different abstraction level, failures could also affect the services participating in the process [SPJ09], thus exercising the fault and compensation handlers specified as part of the benchmark processes [CKLW03]. Thus, it is important to be able to assess how well a given workflow engine can run processes over an unreliable execution environment but also run processes composing services of unreliable providers.

5.8 Monitoring

In addition to process execution, many workflow engines support the monitoring of the process execution by clients. In some case, *live monitoring* allows clients to track in real-time the progress of the process instances they are interested in. In other cases, it is possible to produce periodic monitoring reports, which give some statistics over the number of successfully or unsuccessfully completed process instances in a given time window. Depending on whether the reporting includes data from active processes or it mostly concerns a historical perspective on the instance database, the monitoring feature is likely to have an impact on the performance of workflow-based applications. The challenge is to take into account the impact of monitoring-related features on the process execution benchmarks.

Table 1: Summary: challenges and related performance factors

Logistic Challenges	Performance Factors: 1. Engine Architecture	2. Process Models	3. Web Service Interactions	4. Application Server	5. Instance Database	6. Client Workload
4.1. Collecting Real-World Scenarios		✓				
4.2. Synthesizing Benchmark Flows		✓				
4.3. General vs. Domain Specific Benchmark Flows		✓				
4.4. Benchmark Number and Key Performance Indicators	✓					✓
Technical Challenges						
5.1. Application Implementation Impact Elimination			✓			
5.2. Prevent System Overloading	✓			✓	✓	✓
5.3. Benchmarking Long Running Processes					✓	✓
5.4. System Internal Load Optimization	✓			✓		
5.5. Correlation-based Message Exchange	✓		✓		✓	✓
5.6. Performance Impact of Workflow Language Features	✓	✓				
5.7. Reliability, Recovery and Robustness	✓			✓	✓	
5.8. Monitoring	✓				✓	✓

Likewise, specific monitoring benchmarks can be proposed concerning the performance of the monitoring features of the engine, e.g., how well it scales with a large number of clients that are monitoring a large number of active process instances or retrieving historical reports on very large execution logs.

6 Discussion

The previous two sections present a potentially incomplete list of challenges that developers of a benchmark will face. Our list separates the challenges that are related to the collection of a representative set of usage scenarios (logistic challenges) from the technical challenges that are related to the specific characteristics of WfMS. Incidentally these are the same challenges the developers of WfMSs will face when developing functional

and stress tests of their systems (at least to some extent, since having access to the engine's code makes their life slightly easier). These challenges are also relevant when doing capacity planning to deploy a WfMS in production and enough resources need to be allocated to ensure an acceptable level of performance.

Table 1 summarizes the relationship between the challenges and the performance factors. For each performance factor, we analyzed whether the corresponding challenge has a potential impact on it. The analysis results are based on our experience with the design and architecture of workflow engines, with their performance optimization as well as with initial results of the BenchFlow project [SRF+15].

The logistic challenges mostly relate to finding representative and suitable process models to exercise all capabilities of the workflow engine being benchmarked in a way that is fair and representative of real-world usage scenarios. Attention must be also devoted to the design of a suitable workload model [AW96] that in our case corresponds to the chosen set of process models: lack of an appropriate and scalable workload models can make benchmarking experiments useless [MA01]. Obtaining a single benchmark number out of many possible Key Performance Indicators (KPIs) is influenced by what can be measured considering the engine architecture a black box subjected to a given workload from its clients.

The technical challenges are broadly related to all other performance factors. Eliminating the impact of the external application components and services that are integrated as part of a workflow will require to deal with the way the workflow engine interacts with its Web services. Bringing the system to its saturation point will require to apply a suitable client workload affecting all of its layers (from the database, through the application server middleware all the way to the workflow engine). The benchmarking results obtained with long running processes are likely to be affected by the instance database performance and will require to design specific client workloads. The definition of suitable correlation strategies for the benchmark is a cross-cutting challenge. Concerning the impact of workflow language features, these will need to be exercised by including them in suitable process models. Their performance will be affected by how efficiently they are supported by the workflow engine architecture. Including ways to measure the impact of the fault-tolerance and dependability aspects in the benchmark will need to take into account the effects of the database, the application server and the core of the workflow engine. Likewise, the result of a benchmark targeting the monitoring features will be affected by the organization of the instance database and require to define representative workloads of monitoring clients.

In our work we have begun to address some of these challenges, by making specific design decisions for the benchmark. Due to space limitations we cannot include many details in this paper. However at this point we would like to give a high level description of the approach that we plan to follow.

For challenge 5.1 we have contacted several companies and researchers to ask for process models witout focusing on a specific modelling language. In order to encourage the sharing of the models, we have suggested a) signing of confidentiality agreements and b) implementd a tool for obfuscating and anonymizing processes [SRPL14]. To face challenge 4.2 we analyze the collected models in terms of frequency of use of the BPMN 2.0

features, and frequent structures that are met in the models. With the appropriate usage of this information we plan to implement a workload generator, that will produce the workload according to WfMS specific features. This approach also satisfies challenge 4.3, and partially challenge 5.6 as we can adjust the workload to different features of a WfMS. It is planned that challenge 5.6 will also be addressed with compliance tests of BPMN 2.0 engines to the BPMN 2.0 standard. Challenges 3.2 and 3.3 will be addressed carefully designing a benchmark environment able to handle the diversity and the complexity of BPMN 2.0. Defining the KPIs (Challenge 4.4) will be approached by conducting multiple iterations of the benchmark. In each iteration more complex KPIs will be defined by aggregating metrics obtained with previous iterations and evaluating them based on the community feedback.

7 Conclusion

In this position paper we presented the case for a benchmark for WfMSs that helps compare the performance characteristics of WfMSs and therefore stimulates further research in this important middleware technology. Benchmarks should be simple, portable, scale to different workload sizes, and allow the objective comparisons of competing systems [Gra92]. Considering the large number of factors that impact the performance of a WfMS, the definition of a suitable benchmark remains a challenging open problem. The wide range of successful applications for workflow technology and also the complexity of workflow engines (which potentially interact with many different kinds of middleware) will warrant the exploration of novel approaches to the design of benchmarks specifically targeting the performance of such systems.

By presenting the set of open research challenges collected in this position paper, our goal is to start a discussion within the community interested in middleware for workflow and business process management on the need, potential benefits and possible design approaches for having a set of well-designed and widely accepted benchmarks for assessing, comparing and further improving the performance of WfMSs.

Acknowledgements

This work is partially supported by the Swiss National Science Foundation and the German Research Foundation with the BenchFlow - A Benchmark for WfMSs (DACH Grant Nr. 200021E-145062/1) project.

References

[ACKM04] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures, Applications*. Springer, 2004.

[Act11] Active Endpoints Inc. Assessing ActiveVOS Performance, 2011. `http://www.activevos.com/content/developers/technical_notes/`

assessing_activevos_performance.pdf.

[ALMS09] T. Anstett, F. Leymann, R. Mietzner, and S. Strauch. Towards BPEL in the Cloud: Exploiting Different Delivery Models for the Execution of Business Processes. SERVICES-2 '09, pages 670–677, July 2009.

[AW96] A. Avritzer and E. J. Weyuker. Deriving Workloads for Performance Testing. *Software: Practice and Experience*, 26(6):613–633, 1996.

[BBC+85] D. Bitton, M. Brown, R. Catell, S. Ceri, T. Chou, D. DeWitt, D. Gawlick, H. Garcia-Molina, B. Good, J. Gray, et al. A Measure of Transaction Processing Power. *Datamation*, 31(7):112–118, 1985.

[BBD10a] D. Bianculli, W. Binder, and M. L. Drago. Automated performance assessment for service-oriented middleware: a case study on BPEL engines. WWW '10, pages 141–150, 2010.

[BBD10b] D. Bianculli, W. Binder, and M. L. Drago. SOABench: Performance Evaluation of Service-oriented Middleware Made Easy. ICSE'10, pages 301–302, 2010.

[BCM+05] P. Brebner, E. Cecchet, J. Marguerite, P. Tůma, O. Ciuhandu, B. Dufour, L. Eeckhout, S. Frénot, A. S. Krishna, J. Murphy, et al. Middleware Benchmarking: Approaches, Results, Experiences. *Concurrency and Computation: Practice and Experience*, 17(15):1799–1805, 2005.

[BGNS10] L. Baresi, S. Guinea, O. Nano, and G. Spanoudakis. Comprehensive Monitoring of BPEL Processes. *IEEE Internet Computing*, 14(3):50–57, 2010.

[BKKL09] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing. How is the Weather Tomorrow?: Towards a Benchmark for the Cloud. DBTest '09, pages 9:1–9:6, 2009.

[BSR96] A. J. Bonner, A. Shrufi, and S. Rozen. LabFlow-1: A Database Benchmark for High-Throughput Workflow Management. EDBT '96, pages 463–478, 1996.

[Car07] J. Cardoso. Complexity analysis of BPEL web processes. *Software Process: Improvement and Practice*, 12(1):35–49, 2007.

[Cha95] A. B. Chaudhri. An Annotated Bibliography of Benchmarks for Object Databases. *SIGMOD Rec.*, 24(1):50–57, 1995.

[CKLW03] F. Curbera, R. Khalaf, F. Leymann, and S. Weerawarana. Exception Handling in the BPEL4WS Language. BPM '03, pages 276–290, Eindhoven, The Netherlands, 2003.

[DES08] G. Din, K.-P. Eckert, and I. Schieferdecker. A Workload Model for Benchmarking BPEL Engines. ICSTW '08, pages 356–360. IEEE, 2008.

[DPZ11] F. Daniel, G. Pozzi, and Y. Zhang. Workflow Engine Performance Evaluation by a Black-Box Approach. ICIEIS '11, pages 189–203. Springer, November 2011.

[GGKS02] K. Gottschalk, S. Graham, H. Kreger, and J. Snell. Introduction to Web Services Architecture. *IBM Systems Journal*, 41(2):170–177, 2002.

[GHS95] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and parallel Databases*, 3(2):119–153, 1995.

[GMW00] M. Gillmann, R. Mindermann, and G. Weikum. Benchmarking and Configuration of Workflow Management Systems. CoopIS '00, pages 186–197, 2000.

[Gra92] J. Gray. *The Benchmark Handbook for Database and Transaction Systems*. Morgan Kaufmann, 2nd edition, 1992.

[GSC+04] M. Govindaraju, A. Slominski, K. Chiu, P. Liu, R. v. Engelen, and M. J. Lewis. Toward Characterizing the Performance of SOAP Toolkits. GRID '04, pages 365–372, 2004.

[HHGR06] G. Hackmann, M. Haitjema, C. Gill, and G.-C. Roman. Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices. ICSOC '06, pages 503–508. Springer, 2006.

[HZ06] C. Hentrich and U. Zdun. Patterns for Business Object Model Integration in Process-driven and Service-oriented Architectures. PLoP '06, pages 23:1–23:14, 2006.

[IBM11] IBM SAP. SAP NetWeaver Business Process Management Performance, Scalability, and Stability Proof of Concept. Technical report, IBM SAP International Competence Center (ISICC), Walldorf, Germany, 2011.

[IC07] Intel and Cape Clear. BPEL Scalability and Performance Testing. White paper, 2007.

[Jai91] R. Jain. *Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[JE11] D. Jordan and J. Evdemon. Business Process Model And Notation (BPMN) Version 2.0. Object Management Group, Inc, January 2011.

[KKL06] R. Khalaf, A. Keller, and F. Leymann. Business processes for Web Services: Principles and applications. *IBM Systems Journal*, 45(2):425–446, 2006.

[LBK03] T.-K. Liu, A. Behroozi, and S. Kumaran. A performance model for a business process integration middleware. CEC 2003, pages 191–198, 2003.

[Ley11] F. Leymann. BPEL vs. BPMN 2.0: Should You Care? volume 67 of *BPMN '10*, pages 8–13. Springer, 2011.

[LLHX10] A. Liu, Q. Li, L. Huang, and M. Xiao. Facts: A Framework for Fault-Tolerant Composition of Transactional Web Services. *IEEE Transactions on Services Computing*, 3(1):46–59, 2010.

[LMJ10] G. Li, V. Muthusamy, and H.-A. Jacobsen. A distributed service-oriented architecture for business process execution. *ACM Transactions on the Web*, 4(1):2:1–2:33, January 2010.

[LR97] F. Leymann and D. Roller. Workflow-Based Applications. *IBM Systems Journal*, 36(1):102–123, 1997.

[LR98] F. Leymann and D. Roller. Building a robust workflow management system with persistent queues and stored procedures. ICDE '98, pages 254–258, 1998.

[LRS02] F. Leymann, D. Roller, and M.-T. Schmidt. Web services and business process management. *IBM Systems Journal*, 41(2):198–211, 2002.

[MA01] D. A. Menasce and V. Almeida. *Capacity Planning for Web Services: metrics, models, and methods*. Prentice Hall, 1st edition, 2001.

[MR08] M. Muehlen and J. Recker. How much language is enough? Theoretical and practical use of the business process modeling notation. CAiSE 2008, pages 465–479. Springer, Springer Berlin Heidelberg, 2008.

[Pas05] J. Pasley. How BPEL and SOA Are Changing Web Services Development. *IEEE Internet Computing*, 9(3):60–67, May 2005.

[PF00] M. Poess and C. Floyd. New TPC benchmarks for decision support and web commerce. *SIGMOD Rec.*, 29(4):64–71, December 2000.

[PHA07] C. Pautasso, T. Heinis, and G. Alonso. Autonomic resource provisioning for software business processes. *Information and Software Technology*, 49:65–80, January 2007.

[PP08] P. Pääkkönen and D. Pakkala. Benchmark of middleware protocols for application and service interaction. MUM '08, pages 40–47, 2008.

[Rol13] D. H. Roller. *Throughput Improvements for BPEL Engines: Implementation Techniques and Measurements applied in SWoM*. PhD thesis, University of Stuttgart, 2013.

[RvdAH07] 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.

[SKBB09] 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, 2009.

[SPJ09] S. Stein, T. R. Payne, and N. R. Jennings. Flexible provisioning of web service workflows. *ACM Transactions on Internet Technology*, 9(1):2:1–2:45, February 2009.

[SRF+15] M. Skouradaki, D. H. Roller, L. Frank, V. Ferme, and C. Pautasso. On the Road to Benchmarking BPMN 2.0 Workflow Engines. ICPE '15, 2015.

[SRPL14] M. Skouradaki, D. Roller, C. Pautasso, and F. Leymann. BPELanon: Anonymizing BPEL Processes. ZEUS '14, pages 9–15, 2014.

[Sta10] Standard Performance Evaluation Corporation. SPEC SOA Subcommittee, February 2010. `http://www.spec.org/soa/`.

[Sta11] Standard Performance Evaluation Corporation. SPEC CPU2006 Version 1.2, September 2011.

[Sun07] Sun Microsystems. Benchmarking BPEL Service Engine, 2007. `http://wiki.open-esb.java.net/Wiki.jsp?page=BpelPerformance.html`.

[Tra] Transaction Processing Performance Council. TPC-H. `http://www.tpc.org/tpch/`.

[Tra95] Transaction Processing Performance Council. TPC-D, 1995. `http://www.tpc.org/tpcd/`.

[Tra97] Transaction Processing Council (TPC). TPC Benchmark C (Online Transaction Processing Benchmark) Version 5.11, February 1997.

[TZJW08] A. Traeger, E. Zadok, N. Joukov, and C. P. Wright. A Nine Year Study of File System and Storage Benchmarking. *Transaction Storage*, 4(2):5:1–5:56, 2008.

[WCL+05] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. Ferguson. *Web Services Platform Architecture*. Prentice Hall, March 2005.

[Wes07] M. Weske. *Business Process Management: Concepts, Languages, and Architectures*. Springer, November 2007.

[WFP07] M. Woodside, G. Franks, and D. Petriu. The Future of Software Performance Engineering. FOSE '07, pages 171–187, 2007.

[WLR+09] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann. Monitoring and Analyzing Influential Factors of Business Process Performance. EDOC '09, pages 141–150, 2009.

[WWC92] G. Wiederhold, P. Wegner, and S. Ceri. Towards Megaprogramming: A Paradigm for Component-Based Programming. *Communications of the ACM*, 35(11):89–99, 1992.

[YÖK03] B. B. Yao, M. T. Özsu, and J. Keenleyside. Xbench-a family of benchmarks for XML DBMSs. In *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web*, pages 162–164. Springer, 2003.

[ZTP03] O. Zimmerman, M. Tomlinson, and S. Peuser. *Perspectives on Web Services: Applying SOAP, WSDL, and UDDI to Real-World Projects*. Springer, September 2003.