Evaluating Multi-Tenant Live Migrations Effects on Performance

Guillaume Rosinosky, Chahrazed Labba, Vincenzo Ferme, Samir Youcef, François Charoy, Cesare Pautasso

Evaluating Multi-Tenant Live Migrations Effects on Performance

Guillaume Rosinosky[1 [0000−0001−8980−1231]], Chahrazed Labba[1 [0000−0002−7921−273X]], Vincenzo Ferme[2,3 [0000−0002−0640−6613]], Samir Youcef[1 [0000−0001−7637−7427]], François Charoy[1 [0000−0002−0640−6613]], and Cesare Pautasso[3]

[1] Université de Lorraine, CNRS, Inria, LORIA F-54000 Nancy, France
{guillaume.rosinosky, chahrazed.labba, samir.youcef, francois.charoy}@loria.fr
[2] Reliable Software Systems, University of Stuttgart, Germany
[3] Software Institute, University of Lugano, Switzerland
{vincenzo.ferme, cesare.pautasso}@usi.ch

Abstract. Multitenancy is an important feature for all Everything as a Service providers like Business Process Management as a Service. It allows to reduce the cost of the infrastructure since multiple tenants share the same service instances. However, tenants have dynamic workloads. The resource they share may not be sufficient at some point in time. It may require Cloud resource (re-)configurations to ensure a given Quality of Service. Tenants should be migrated without stopping the service from a configuration to another to meet their needs while minimizing operational costs on the provider side. Live migrations reveal many challenges: service interruption must be minimized and the impact on co-tenants should be minimal. In this paper, we investigate live tenants migrations duration and its effects on the migrated tenants as well as the co-located ones. To do so, we propose a generic approach to measure these effects for multi-tenant Software as a Service. Further, we propose a testing framework to simulate workloads, and observe the impact of live migrations on Business Process Management Systems. The experimental results highlight the efficiency of our approach and show that migration time depends on the size of data that have to be transferred and that the effects on co-located tenants should not be neglected.

Keywords: Live migration · Multitenancy · BPMS · Performance

1 Introduction

A software service provider that wants to provide its service in the Cloud needs to adjust the Cloud resources it consumes to the needs of his customers. A Web based business application requires the deployment of a software stack including application servers and databases. In order to accommodate multiple customers, the provider can rely on a multi-tenant architecture that allows managing several

customers on the same server. Thus, to accommodate all customers with the minimal set of resources, it might be necessary to «move» tenants from one installation to another one. During peak hours, many resources might be required, while at night a single resource could respond to all the requests from all customers. *Live migrations* can be used to ensure such kind of optimization. It can lead to significant gains in the operational costs. Migrating a tenant from a ressource to another has consequences. It takes time during which the service is not available for the customer. It consumes resources as CPU, networking, and disk IO that may affect the Quality of Service (QoS) for other co-located customers. In this work, we investigate the effects of multi-tenant Web applications live migrations. To the best of our knowledge, our work is the first to focus on evaluating the impact of live migrations on both service interruption time and co-located tenants performances for a service including both application and database servers. To investigate the effects of tenants live migration, we propose an approach to evaluate: **(i)** the duration of service interruption when a tenant is migrated; **(ii)** the effects of a live migration on the migrated tenant; and **(iii)** the effects on the performances of other tenants hosted on both origin and destination resources during the migration. The proposed approach is generic and can be applied for all multi-tenant Web applications sharing a set of resources (computing and/or storage). We apply it to a multi-tenant Business Process Management System (BPMS), representing a classical transactional application. To do so, we propose a performance test framework to investigate the efficiency of our approach in determining the effects of migrations for multi-tenant BPMS. The main design goal of the proposed framework is to create a dynamic testing environment that scales in terms of resources and tenants numbers.

To summarize, the main contributions of this paper are as follows:

- A generic approach to measure the impact of the live migrations in terms of service interruption time and performances (execution time) for multi-tenant Web applications. (Section 3)
- A performance test framework for multi-tenant BPMS. Indeed a detailed description of the framework architecture is presented. (Section 4)
- An example on how the framework can be used for measuring the effects of live migrations on the Performance edition of the BPMS Bonita 7.4.3[4]. (Section 5)

The rest of the paper is organized as follows. Section 2 describes a motivating example that we use throughout this paper to explain the different steps of the proposed method. Our proposed approach for evaluating migrations effects is introduced in section 3. The test framework for multi-tenant BPMS is presented in section 4. Experiments and empirical results are presented in Sections 5 and 6. Related work, Threats to Validity and Conclusion and Future Work are presented respectively in Sections 7, 8, and 9.

[4] http://www.bonitasoft.com

2 Motivating Example

Live migrations of tenants is important for the service provider. Let's consider a Business Process Management as a Service setting consisting of a BPMS deployed on a Cloud provider infrastructure. The Cloud provider offers a set of computing resources. For performance reasons, a BPMS installation is deployed on a resource that consists of at least two compute instances for both a database and an application server. Each resource is characterized by its price per unit of time as well as a minimum and a maximum tasks throughput per second. Each active BPMS instance is used by customers named tenants. Each tenant has a QoS requirement expressed in terms of BPM task throughput. In Figure 1, the tenant $T4$ has an initial BPM task throughput of (40 task/second) at time (t=$t0$). It goes up to (60 tasks/second) at time (t=$t1$). The widths of the resources and the tenants correspond respectively to the capacity and the requirements in terms of task throughput.

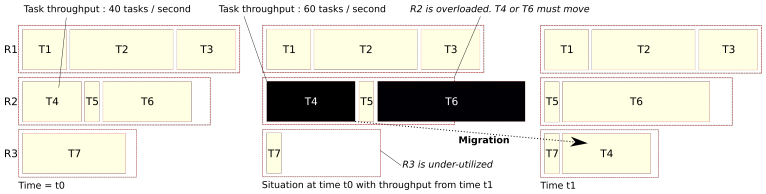

Fig. 1. Tenants Distribution at Time $t = t_0$ and $t = t_1$

At time (t=$t0$), the tenants are placed on the different resources. For example, in Figure 1, the tenants $T1$, $T2$ and $T3$ share the first installation deployed on the resource $R1$. At time (t=$t1$), the tasks throughput of the different tenants (presented in black) changes, which leads to over and under utilization, respectively of $R2$ and $R3$. We reorganize the distribution of tenants on the different resources. An appropriate solution consists in migrating $T4$ to $R3$. Migrations may induce a service disruption on the tenant side and is not instantaneous. The interruption time for $T4$, includes its deactivation on the origin installation $R2$, its migration to $R3$, and its activation on $R3$. Further, it may have an impact on the performances of the co-located tenants. Thus, we are interested in providing a way to investigate the effects of the migrations on the service interruption time and the performances of the co-located tenants. The next section presents the proposed approach used to solve the aforementioned challenge.

3 Measuring the Impact of Migration

In this section we present our approach to evaluate the impacts of tenants live migrations on both service time interruption and tenants performances. First, we present the metrics we use to detect these impacts. After this, we describe

the assumptions we make related to the effects of live migrations. Then, we explain the generic setup we propose to do the measurements in a way that can be replicated for Web applications including application and database servers.

To the best of our knowledge, there is no zero-downtime technique to perform tenants migrations from an origin to a destination resource. Further, with regular relational databases, there exist no way to perform live migrations from one resource to another, with a zero-downtime for the tenant. During a migration, the tenant may undergo a limited availability of the services provided by the Web application, causing QoS breaks. We characterize the impact of tenants migrations on the overall software operations, using three metrics.

The **migration duration** or service interruption time is the time that can be measured between the beginning of the migration on origin resource, when it stops accepting requests and the time when it starts accepting requests on the destination installation.

The second and third metric measures the impact of migrations on performances of tenants hosted on origin and destination resources. We measure the processing time for the migrated tenant and for co-located ones on the origin and destination resources, this during, and after the migration. We name these two metrics ***migration effects on migrated tenant***, and ***migration effects on co-located tenants***.

We want to measure the processing time of the executing operations from the customer and the service provider perspectives. Fulfilling a client request does not imply the termination of the associated operations on the service provider side. A BPMS engine may take additional time to terminate the operation after satisfying the client request as there could be long executing processes running after the answer to the client.

We have several assumptions about the negative effects of migrations on the service interruption time and the performances, that we want to verify and measure using our approach :

- *Migrations duration* is predictable and increases with the size of the data. Migrations consist of copying Web application data from the origin resource to the destination resource. More data means more time to copy.
- *Migration effects on migrated tenant* exist and are provoked by the creation of a new tenant on the destination installation. This could have various causes such as negative cache hit or asynchronous libraries initialization.
- *Migration effects on co-located tenants* exist and must be taken into account. The live migration will have effects on both the origin and destination resource of the migration, and their tenants. As tenants are activated, deactivated, and their data is being read, copied across the network and written on the resources, it may induce performance degradation on both resources executing co-located tenants operations.

In order to measure these metrics, we simulate users interactions with the studied Web application. For each measurement, we observe variations such as the nature of the workload and the quantity of operations initiated in the system.

We store every duration of each launched operation on the Web application as well as of each migration.

We propose an experiment where we first execute a defined workload for a tenant, hosted on the origin resource. We then migrate this tenant from the origin resource to the destination resource, and observe the corresponding migration duration. For each experiment, we store each corresponding time stamp, including the relevant internal steps (such as the beginning and end of the movement of data, the eventual phases of deactivation and reactivation of tenants, etc.).

In order to evaluate the *Migration effects on migrated tenant*, we measure the performances of the tenant before migration on the origin resource, and after migration on the destination resource. We assume we can retrieve Web application response times, if possible. This is usually stored inside the database of the Web application. It should also be possible to retrieve these metrics from the load testing tool point of view. All the duration of the HTTP queries and corresponding processing durations are timestamped and fully identified so we can compare them with the point of view of the application.

We want to compare the effects of a migration on the other tenants currently hosted on the origin and destination resources. In this case, we propose to initiate workloads on tenants on both origin and destination resource. We also initiate a shorter workload for a tenant on the origin resource who will be migrated after to the destination resource, as in previous experiments. A workload is then executed on the migrated tenant, after the migration.

The *Migration effects on co-located tenants* are retrieved for the co-located tenants hosted on both origin and destination resources. In order to do this one should retrieve response times of the HTTP queries and corresponding processing durations for tenants. We consider as “query concerned by migration" every query running between the beginning of the migration and the end of the migration, i.e., where the timestamp of the beginning of the query is before the end of the migration, and the end timestamp of the query is after the beginning of the migration. We then compare to other processing durations, when there is the same number of tenants, with the same load and on the same resources. The differences in processing durations will show the effects of the migration on the co-located tenants.

4 Experiment Framework

In order to validate our approach, we developed a testing framework to study the impact of live migrations for multi-tenant BPMS. According to the type of the performance metric to be investigated, the framework takes as input a set of parameters and provides as output a set of measurements. The Figure 2 depicts the architecture of the proposed framework. We distinguish the following elements:

– The migration method used to move the tenants from one installation to another (out of the scope of this paper).

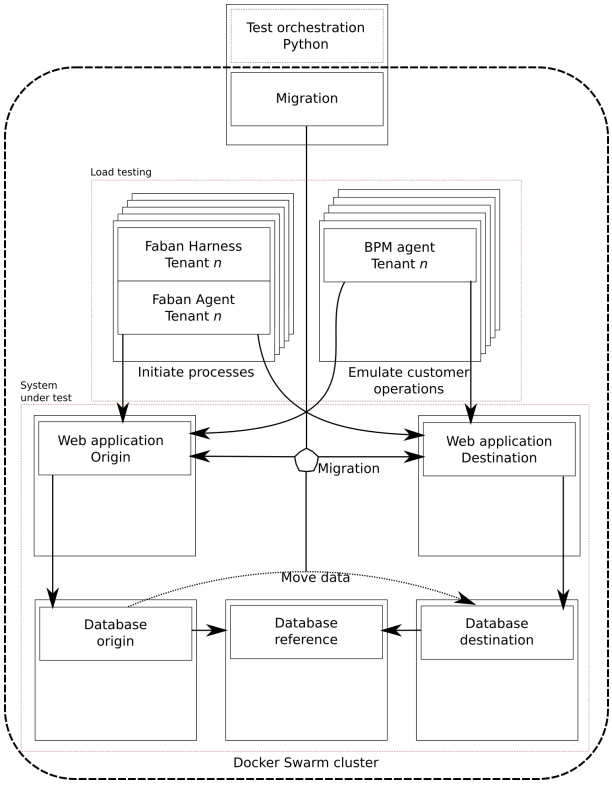

Fig. 2. The framework architecture for studying the impact of live migrations on multi-tenant BPMS

– The load testing components used to generate variable workloads to the System Under Test (SUT).
– The SUT represents the BPMS to be studied.

In the rest of this section, we present a detailed description of each component as well as the orchestration of the framework to perform the measurements.

The load testing components (Figure 2) emulates the behaviour of users within tenants interacting with the BPMS.

In the case of a BPMS, we used two different categories of load testing components : a load tester for tenant and processes initialization and BPM agents for BPM tasks retrieval and processing.

The goal of the load tester is the following:

– Initialize all the tenants as well as the users within each one.
– Deploy the BPM processes needed for the experiments, and add the authorizations and roles needed for the proper functioning of the BPMS.
– Start process instances according to the specific test requirements, by initializing calls to the HTTP API of the BPMS.

The load tester is implemented relying on the Faban framework[5] used for the BenchFlow framework [3]. It enables the specification of complex workload, as well as the number of simulated users that are interacting with the SUT, and the duration of the interaction. The Faban framework then takes care of executing the specified load test, and ensure a correct and verified execution of the defined load.

A BPMS tenant has an organizational structure that consists of different groups of users. According to their profiles and roles, users carry out different business activities within the organization. We use a multi agent system (MAS) to model and simulate the users involved in the execution of business activities within each tenant. A MAS consists of a number of **agents** in a common environment (real/virtual) where they can act and cooperate to achieve system objectives [10]. In our work, we focus on the mapping of the organizational settings within tenants to agents. Using agents on behalf of tenants users provides a suitable means of reproducing human behaviour. Due to their distinct capabilities in terms of autonomy, flexibility, and adaptability agents present effective solutions in modeling and simulating behaviours of human resources. We use agents to imitate human resources in order to: (i) work with process instances including various proportions of human tasks, which are similar to those deployed within real organizations; and (ii) maintain a given threshold of active tasks during tenants migrations to investigate its impact on the migration duration as well as on the co-localized tenants on both origin and target installations.

Each agent represents an active entity that performs specific tasks on behalf of a user within a BPMS tenant. The agent user behaviour consists of the following actions: the agent first connects to the BPMS platform. Then, it starts retrieving the available tasks. If the number of ready tasks is under a given threshold, which represents the sought tasks number before tenant migration, the agent waits until more process instances are started. Else, the agent executes the task after assigning it to itself.

In order to fulfill reproducibility, our framework uses Docker containers. Docker permits easy operating-system-level virtualization. We use it to the launch of the SUT, and for the injection and migration scripts and tools. We used Docker Swarm[6] for our cluster management, and Docker Compose for the description of the distributed system. Swarm is a functionality of Docker to manage containers on several nodes. Docker Compose allows to describe a complete stack, which can be composed by several containers having dependencies between them, as the use of the database by the Web application.

We also used Faban and BPM agent Docker Compose files in order to launch them easily, with all the required parameters such as the the names of the users, tenants, and processes.

To use the framework, we must prepare a test descriptor containing the various parameters of the experiments. The parameters can be customized in

[5] http://faban.org/about.html
[6] https://docs.docker.com/engine/swarm/

order to evaluate the behaviour depending on the BPM schema, the BPMS, the duration of the BPM processes injection, etc.

Once a test is launched, the following steps are triggered:

1. Test initialization: creation of unique identifier for the test and experiment directory.
2. Deploy initialization: reset and initialize Docker Swarm on the resources and needed files for the experiment.
3. Deploy BPMS: deploy the BPMS and corresponding databases containers on the resources.
4. Launch test: deploy the Faban load testers, the BPM agents on the concerned resources, and execute the tests for the concerned tenants.
5. Copy results: launches queries on BPMS database in order to get processing time, retrieve results and store them.

In step 4 we launch multiple loads on a set of tenants for a defined duration, named "background tenants" while a loaded tenant is migrated. The tests for the background tenants are launched in an asynchronous manner. A launch of the Faban load tester coupled to the BPM agent is triggered on the studied tenant ("migrated tenant") for a short duration. Once this first step is finished, the framework waits for a defined duration to let all processes finish. Then the migration is triggered. After this migration, a second process injection is launched on the destination resource. Results of the tests can be then retrieved from the BPMS, the BPM agents and the Faban agents.

In step 5, we retrieve the measures. The results consist in the duration of the migration, the response time and the duration of processing of each process and tasks launched during the test. These results have a timestamp stored in CSV file with the identifier of the BPM task and its corresponding process, and their durations.

5 Experimentation

We explain in this section how we have adapted our framework[7] for our use case. As shown in the Figure 2, the SUT is composed of five software elements,including the origin and destination stacks consisting of one BPMS engine and its corresponding database, and a reference database whose goal is to host the archive and system data.

For our tests, we use BonitaBPM 7.4.3[8] in its commercial Performance edition[9].

We conducted these tests with different parameters for the origin and target resources, and for multiple processes quantities. We needed a method to migrate tenants, a realistic workload, test scenarios, and test infrastructure. All the experiments characteristics are described in the rest of this section.

[7] https://github.com/guillaumerosinosky/migration_bpms
[8] http://www.bonitasoft.com
[9] Multi-tenancy is only available in this commercial licence.

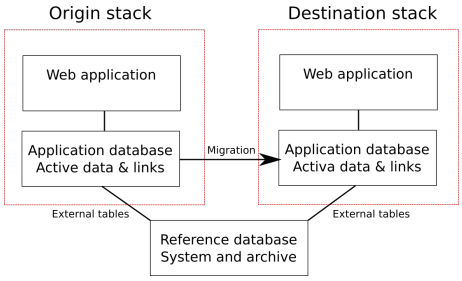

Fig. 3. Proposed database architecture. Software uses a relational database. A “reference" database for system-related and archive data is used. External tables are used for the links between the reference database and the database used by the Web application.

Live migration of tenants requires a series of steps. BonitaBPM allows stopping and restarting the tenant operations. When a tenant or his underlying operations are in a stopped state, new operations cannot be launched, and current operations are put on hold.

We have used specific Docker Compose file referencing two Bonita Docker containers, two Postgresql 10 Docker containers for the application databases, and one for the reference database. Figure 3 shows the corresponding five containers, which will be generated for each experiment.

We use three different business processes models to investigate the impact of the different workflow structures on the migration duration. The processes are modeled using the graphical standard language BPMN[10]. The first model consists of a single human task.The other models represent a combination of human and automatic tasks as well as parallel and exclusive gateways. These models were initially defined and used in [4]. The second model, in Figure 4, consists of a set of automatic tasks implemented as script tasks and a single human task. The process starts by initializing an integer number ($x = 0$), which is incremented by the script task situated after the first exclusive gateway. With respect to the value of the variable (x), the upper and lower branches are followed. The model, shown in Figure 5, has a more complex structure compared to the processes models above. It consists of two human tasks executed in parallel. The process starts by initializing an integer number ($x = 0$), which will be later incremented within the script task situated after the gateway "endAsk". With respect to the value of the variable (x), the upper and lower branches are followed. To have a deterministic behaviour, the upper path, which contains the last exclusive gateway is executed till ($x == 4$) before the ending of the process.

In order to evaluate the three metrics presented in section 3, we have conducted two sets of experiments:

– The first experiment evaluates the *Migration duration*. In this case, we have launched the framework for only one tenant. We executed the load on the

[10] Business Process Modeling Notation

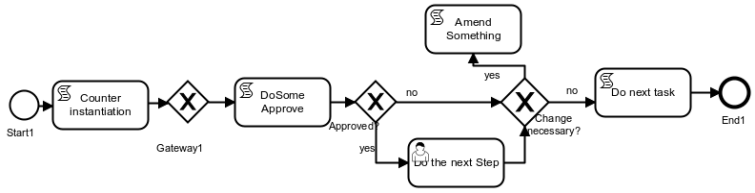

Fig. 4. AdditionalApproval BPMN schema

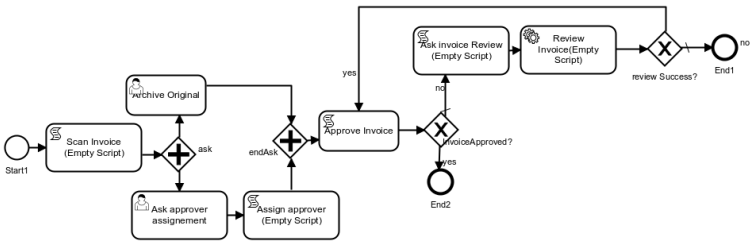

Fig. 5. M3Process BPMN schema

origin resource for iteratively 0, 5, 10, 30, 60, 120, 300, 600 seconds for the three types of BPM schemas, and then we performed three migrations of the processes from the origin resource to the destination resource, and vice versa. We observed the total duration of the migration, including its different steps: deactivation of tenant, migration of the data and activation of the tenant.
– The second experiment observes *Migration effects on migrated tenant* and *Migration effects on co-located tenants* tests. In this case, we have three tenants: one background tenant running on the origin resource (*tenant1*), one background tenant running on the destination resource (*tenant3*). Each one has one Faban agent injecting processes for 20 minutes, while 100 BPM agents close their tasks. A third tenant, tenant2, is launched first for 2 minutes on the origin resource. The framework waits for 5 minutes in order to let every process finish, and to have comparison for Test 3. Then a migration from the origin resource to the destination resource is triggered, and a 2 minutes load is launched for this tenant on the destination resource.

We have conducted the tests on Azure Public Cloud. We used the following instances types:

– Databases: Standard E2s v3 (2 vcpus, 16 GB memory) - 3 instances
– BPMS: Standard F4s (4 vcpus, 8 GB memory) - 2 instances
– Faban load tester (Harness and Agent): Standard F1s (1 vcpus, 2 GB memory) - 1 or 3 instances respectively for the first and second experiment
– BPM Agents: Standard F2s (2 vcpus, 4 GB memory) - 0 or 3 instances respectively for the first and second experiment
– Orchestrator: Standard B2ms (2 vcpus, 8 GB memory) - 1 instance

E-series are memory-optimized instances[11]. We used it for the database instances. F-series[12] are computing-optimized instances. We have chosen it for the BPMS, the BPM agent and the Faban load tester. The instance we have used for the orchestration and data collection is a small size burstable instance.

6 Empirical Results and Analysis

In this section, we present the results we obtained following our experimentation. Due to space limitations, detailed results are shared here[13]

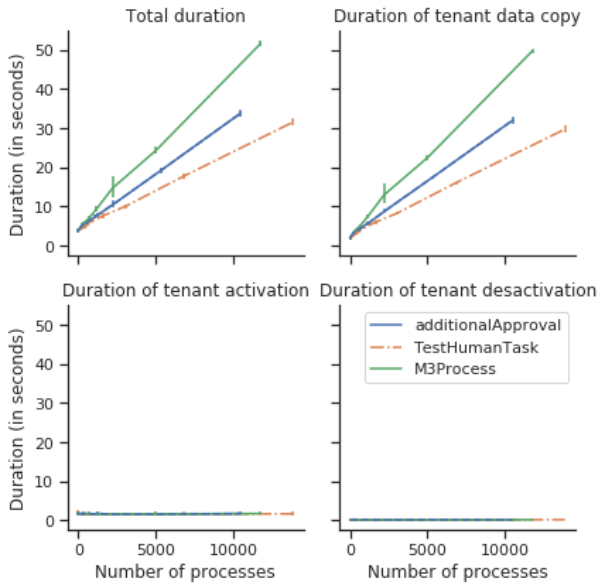

Fig. 6. Mean duration vs number of processes

Figure 6 shows the duration for: the deactivation of the tenant on the origin resource, the migration of the data, the activation on the destination resource, and the total duration of the migration.

The deactivation of tenants is very fast (less than 1 second) and stable regardless the number of active processes or the BPM schema. The activation of tenants is less stable, and last for a few seconds. The copy of the data seems very

[11] Standard memory optimized instance: https://docs.microsoft.com/en-us/azure/virtual-machines/linux/sizes-memory

[12] Standard computing optimized instance: https://docs.microsoft.com/en-us/azure/virtual-machines/linux/sizes-compute

[13] http://dx.doi.org/10.5281/zenodo.1402632

linear. It is the fastest for the smaller schema, TestHumanTask (about 20 seconds for 10000 processes), then longer for the schema addtionalApproval (more than 30 seconds), and even longer for the M3Process schema. All durations stay the same for 0 processes, and there is some variability, which may be linked to the uncertainties of Cloud behaviour.

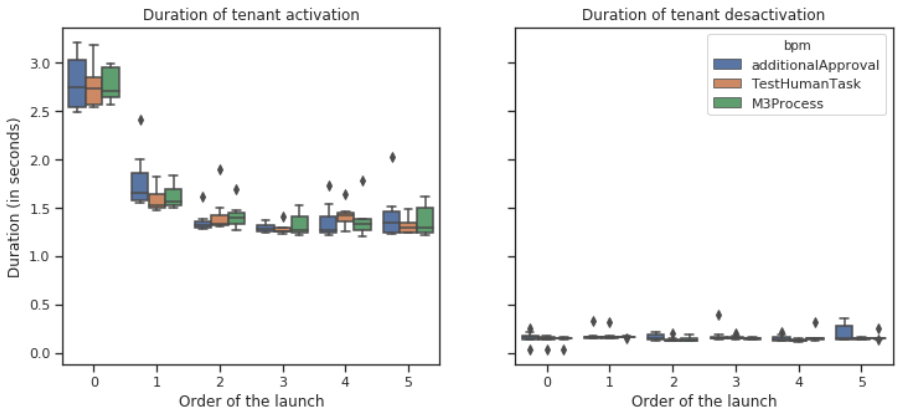

Fig. 7. Duration distribution vs order of launch

Figure 7 provides details on the duration of the deactivation and activation of tenants compared to the number of invocation. The duration is stable for the deactivation, between 0.05 and 0.4 seconds for all the experiments. This is not the case for the activation corresponding to the first migration. It always last for more than 3 seconds when most other last around 1.5 seconds. The second migration has a higher duration than the rest. Apart of some longer experiments, which are occasional, the behaviour of both activation and deactivation is similar regardless the BPM schema, and the number of migrated processes.

Duration vary between 5 seconds for no active processes to about 30 seconds for about 10 thousands processes for the TestHumanTask processes. The process comprised of a parallel tasks has a longer duration: 50 seconds for more than 10000 injected processes, this is about 20 seconds more than for sequential processes. This is probably caused by the migration of two active tasks instead of only one for other processes. This shows that in order to evaluate the duration of live migrations, one must consider the BPM schema structure. The duration of the migration of the schema M3process is also a bit longer than the duration of the HumanTask. The presence of numerous automated tasks and gateways could provoke the storage of additional data for this process.

We have seen in the last part that the duration of the activation part of the migrations is longer for the first migrations. The reader must remember that in this experimentation we have launched iteratively 6 migrations, 3 from the origin to the destination and 3 for the destination to the origin, using the same

tenant. The probable explanation here is that the first activation of the tenant in a resource needs some initialization in the tenant's metadata (such as library, cache initialization, filesystem based resources, etc.), which do not exist yet in the filesystem or memory of the target resource's installation when it is migrated for the first time . This behaviour happens only one time, and the duration of the activation part stays stable after this first.

Now, we present the results for processes durations on the origin resource, and on the destination resource of a migrated tenant. The Table 1 shows the general statistics depending on the observed schema, and when the results have been retrieved in the experiment(before -pre- or after -post- the migration). The mean duration of TestHumanTask process is about 1300 milliseconds shorter after the migration compared to before, but 1500 milliseconds longer for the M3Process schema and 2700 milliseconds for the AdditionalApproval process. The standard deviations are similar except for the TestHumanTask process.

Table 1. Duration of tasks grouped by BPM schema and moment compared to the migration.

BPM schema	When	Count	Mean	Std	Min	25%	50%	75%	Max
M3Process	post	10292.0	38449.2	16217.7	13657.0	26046.0	34401.0	44808.0	76321.0
M3Process	pre	14630.0	36950.8	15909.0	4398.0	24837.0	33933.5	47169.0	87303.0
TestHumanTask	post	19529.0	12579.3	9378.1	581.0	3081.0	9588.0	15921.0	32269.0
TestHumanTask	pre	19688.0	13892.8	15594.2	466.0	2969.5	12282.0	19553.5	168560.0
AdditionalApproval	post	17710.0	113370.0	42819.4	22915.0	78601.0	116355.0	146674.0	199938.0
AdditionalApproval	pre	25749.0	110685.8	42710.7	12907.0	83799.0	114749.0	141261.0	199856.0

We obtain slower mean durations for M3Process and additionalApproval processes after the live migration, and faster mean duration for the TestHumanTask process. At the time of writing, we don't have rational explanation for that.

We compare in figure 8 the duration of processes from the BPM provider perspective (left), and task duration from the BPM agent perspective (right). On both figures, the blue (left) boxplot for each number of active processes is the duration when no migration is executed, and the orange (right) boxplot is the duration during the migration.

There are a lot of variation probably due to the Cloud behaviour, and to some non deterministic effects after the live migrations. Some processes last for more than 200 seconds, up to 1000 seconds. Even with these effects, the duration of both processes and tasks are positively correlated to the number of active processes. Durations are a bit longer and less stable during migrations.

In order to have a more homogeneous comparison, we removed from the data the processes having a duration superior to 200 seconds and tasks superior to 15 seconds. The table 2 shows the corresponding results for the BPM tasks response time, aggregated by tenant. The table shows that the duration is always longer during migrations, both for the tenant on the origin resource (tenant1) and on the destination resource (tenant3). The difference in duration is between 26 and 650 milliseconds. We cannot find correlation with the BPM schema.

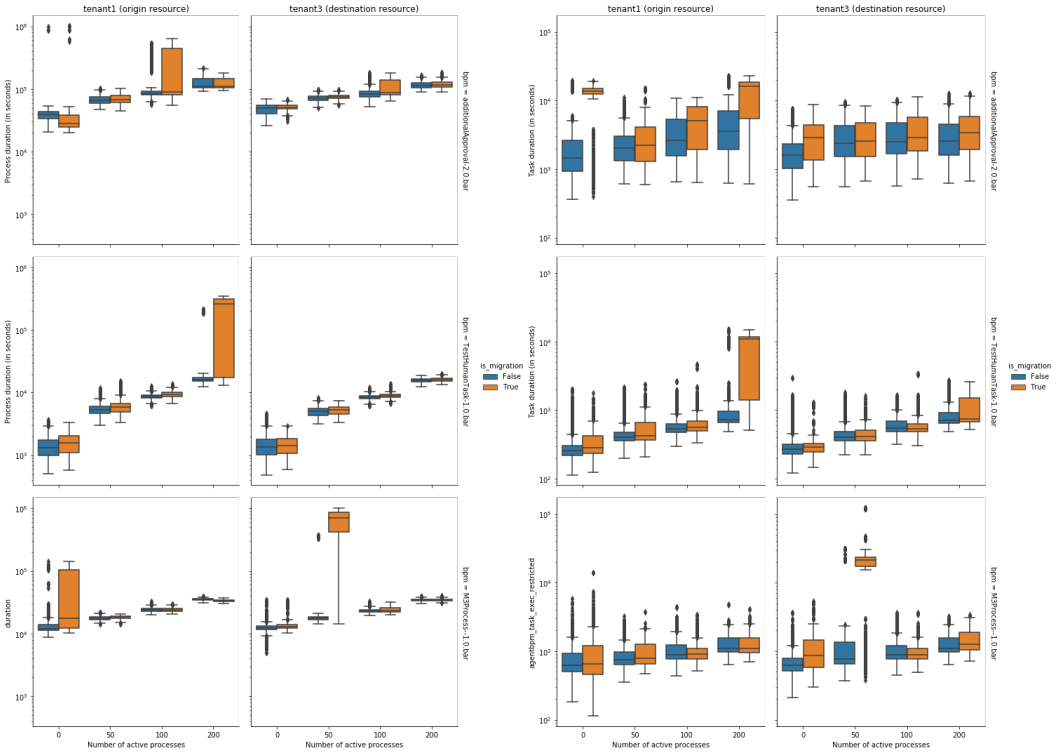

Fig. 8. Process duration -BPMS point of view- distribution (left) and task duration -BPM agent point of view- distribution (right) vs process schema and number of active processes.

The performances of tenant1 and tenant3 are similar, sometimes faster, and sometimes slower.

As in the previous experiments, additional process is much longer than M3Process, which is longer than human task. The same behaviour occurs for the composing tasks at a much smaller scale.

The effects of the live migrations on the co-located tenants are not important but they exist. We observe a few hundred more milliseconds for each type of BPM schema. It is higher for the more complex BPM schemas than for the simpler. This is the same when we compare the effects on the tenant of the origin resource to those on the destination resource. Migrations have more effects on the AdditionalApproval process destination resource than on the origin resource. This will require more investigations. The results we obtain are similar for the process duration.

Table 2. Duration of tasks during migration compared to before grouped by BPM schema and tenant

BPM schema	Tenant	Migration running	Count	Mean	Std	Min	25%	50%	75%	Max
M3Process	tenant1	False	11745.0	1018.9	511.4	183.0	668.0	884.0	1157.00	5819.0
M3Process	tenant1	True	4602.0	1126.4	695.8	115.0	738.0	962.0	1302.75	13926.0
M3Process	tenant3	False	11351.0	1003.2	475.9	208.0	674.0	874.0	1174.00	3612.0
M3Process	tenant3	True	3995.0	1208.9	600.1	297.0	795.0	1025.0	1533.50	5270.0
TestHumanTask	tenant1	False	17375.0	718.0	1171.1	114.0	330.0	482.0	687.00	11671.0
TestHumanTask	tenant1	True	4107.0	734.1	464.6	124.0	439.0	617.0	791.50	4665.0
TestHumanTask	tenant3	False	18857.0	609.5	393.7	122.0	345.0	505.0	704.00	2959.0
TestHumanTask	tenant3	True	4667.0	738.6	491.7	145.0	426.5	603.0	791.00	3444.0
AdditionalApproval	tenant1	False	15131.0	2790.5	2055.7	360.0	1281.0	2100.0	3404.00	12241.0
AdditionalApproval	tenant1	True	5405.0	2883.6	2359.1	398.0	1313.0	2152.0	3484.00	14949.0
AdditionalApproval	tenant3	False	19530.0	3050.3	2109.8	354.0	1458.0	2255.5	4110.00	12673.0
AdditionalApproval	tenant3	True	8299.0	3702.0	2349.8	558.0	1751.0	2935.0	5521.00	12882.0

7 Related Work

Performance is a major concern for multi-tenant service providers. In the literature, performance evaluation for multi-tenant environments is tackled from different perspectives including evaluation of tenant placement [12][11], live migrations [5][2] and the suitability of the sharing approach [6][8]. The target applications are mainly multi-tenant databases, whereas in our work, we focus on evaluating the tenants migration effects for a Web application including both application and database servers. Initially, the live migration is used for Virtual Machines [1]. The approach consists in moving a virtual machine and its dependencies from one hardware resource to another without being shutdown. The aim is to decommission the hosting of resources, which are not needed anymore, while minimizing the downtime and the negative effects. In [6], the authors extend the TCP application for benchmark to support multi-tenant platforms. The performance evaluations focus on determining the maximum throughput as well as the tenants number supported by the platform, which can be used later to provide insights about the suitable sharing approach. Although, in our work, the number of the tenants as well as their requirements in terms of throughput are known in advance, our main focus is on investigating tenants live migrations impacts on duration and response time, which is not supported in [6]. In [12], the authors present the STeP framework for scheduling multi-tenant databases on the suitable resources. SteP provides a new set of tenants packing algorithms to optimize both static and dynamic resource allocation. Further, it uses a set of metrics including performance objective violation, the operations cost and the monetary penalty to determine the efficiency of the placement approach. In our work, we focus on the impact of the dynamic placement and its effects on the tenants where the authors consider only scheduling. In [9], the authors present a profit-driven tenant placement strategy for multi-tenant databases. Similar to [12], a set of placement algorithms are proposed, which were evaluated based on operations cost and the SLA penalty costs. Although the authors discuss

the suitable deployment of multiple tenants, the migration of tenants and its impacts are kept as a future extension of their work. In [7], the authors propose a framework that furnishes policies for hardware provisioning and tenants scheduling according to the tenants classes and their performance SLOs. While in our work, we assume we know in advance the required resources to deal with the evolving requirements of tenants and we provide an approach to measure the impact of the live migration in terms of duration and response time.

8 Threats to Validity

The proposed approach has different limits, that we tried to mitigate when possible:

- We have tested the effects of co-located tenants only for small numbers of active processes (0 to 200). We need to test with more active processes.
- Using Public Cloud resources induces variability problems. As we have seen in the results, even when executing multiple tests we still have many outliers, sometimes on a whole experiment. The only solution is to launch even more experiments. We plan to make a more intensive analysis of migration for a next iteration, and the framework will help us in this task.
- We have not studied the effects of multitenancy on the performance. In this case we have compared results for a fixed number of tenants. Indeed, there could be additional operations related to the management of tenants, but non related to the workload. Response time could not be totally proportional to the injection if there are more tenants. We plan to study this in a future work.
- The results concern only one BPMS and one live migration method. We plan to test with more BPMS from different vendors.
- In its current implementation, the framework scales in the number of Faban/BPM agents. However, when we tried to scale up the number of Faban load tester instances, we had issues with the parameters. The SUT should to be tuned in order to take advantage of this (resource-wise, and parameter-wise). In this experiment, we have tuned the SUT in the same way for all the experiments. We plan to execute tests with different configurations of the parameters and test different resource configurations in future work.

9 Conclusion and Future Work

In this paper, we presented two contributions: (i) a generic approach to measure the impact of live migrations in terms of service interruption and performances evaluation for Web applications; and (ii) a performance test framework for multi-tenant BPMS. We present an example on how the framework can be used for measuring the effects of live migrations using a well known BPMS. The analysis of the experimental results showed that the duration of a migration depends

linearly on the size of the active data and that the effects on the co-located tenants cannot be neglected. Indeed, live migrations may last for long and providers should take them into account and study their effects before on their applications if they want to guarantee the same QoS. This fact could included in the migration decision of elasticity algorithms. Further, our framework can be easily used for other Web applications or other live migration methods with the appropriate adaptation mentioned in the paper.

As a future work, we plan to perform deeper analysis on the effects of live migrations on co-located tenants, for various number of tenants, live migration methods, and Web applications in order to have better view on their QoS impacts. We also intend to enhance our framework, by improving the BPM agent component through the modeling and simulation of more advanced human behaviours. Our work allows to have a better view on live-migrations effects and pinpoint important criteria Software as a Service (SaaS) providers should not underestimate. It can be used to evaluate deployment decision and to parameterize elasticity algorithms when QoS constraints are very strong.

Acknowledgments

This work has been partly supported by the German Research Foundation (HO 5721/1-1, DECLARE), and by the Swiss National Science Foundation (project no. 178653). This work has been supported by Azure Research Grant. We thank heartfully Bonitasoft without whom this analysis could not have been done.

References

1. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2. pp. 273–286. USENIX Association (2005)
2. Das, S., Nishimura, S., Agrawal, D., El Abbadi, A.: Albatross: Lightweight elasticity in shared storage databases for the cloud using live data migration. Proc. VLDB Endow. **4**(8), 494–505 (May 2011)
3. Ferme, V., Ivanchikj, A., Pautasso, C.: A framework for benchmarking bpmn 2.0 workflow management systems. In: International Conference on Business Process Management. pp. 251–259. Springer (2015)
4. Ferme, V., Ivanchikj, A., Pautasso, C.: Estimating the cost for executing business processes in the cloud. In: International Conference on Business Process Management. pp. 72–88. Springer (2016)
5. J. Elmore, A., Das, S., Agrawal, D., El Abbadi, A.: Zephyr: Live migration in shared nothing databases for elastic cloud platforms. In: SIGMOD. ACM, ACM (06/2011 2011)
6. Krebs, R., Wert, A., Kounev, S.: Multi-tenancy performance benchmark for web application platforms. In: Daniel, F., Dolog, P., Li, Q. (eds.) Web Engineering. pp. 424–438. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
7. Lang, W., Shankar, S., Patel, J.M., Kalhan, A.: Towards multi-tenant performance slos. In: 2012 IEEE 28th International Conference on Data Engineering. pp. 702–713 (April 2012)

8. Liu, R., Aboulnaga, A., Salem, K.: Dax: A widely distributed multitenant storage service for dbms hosting. Proc. VLDB Endow. **6**(4), 253–264 (Feb 2013)
9. Liu, Z., Hacigümüs, H., Moon, H.J., Chi, Y., Hsiung, W.P.: Pmax: tenant placement in multitenant databases for profit maximization. In: EDBT (2013)
10. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education, 2 edn. (2003)
11. Schaffner, J., Januschowski, T., Kercher, M., Kraska, T., Plattner, H., Franklin, M.J., Jacobs, D.: Rtp: robust tenant placement for elastic in-memory database clusters. In: SIGMOD Conference (2013)
12. Taft, R., Lang, W., Duggan, J., Elmore, A.J., Stonebraker, M., DeWitt, D.: Step: Scalable tenant placement for managing database-as-a-service deployments. In: Proceedings of the Seventh ACM Symposium on Cloud Computing. pp. 388–400. SoCC '16, ACM, New York, NY, USA (2016)