Performance Assessment of Cloud Migrations from Network and Application Point of View

Lukas Iffländer[1], Christopher Metter[2], Florian Wamser[2], Phuoc Tran-Gia[2], and Samuel Kounev[1]

[1] Chair of Software Engineering, University of Würzburg, Würzburg, Germany
{ifflaender|kounev}@informatik.uni-wuerzburg.de
[2] Chair of Communication Networks, University of Würzburg, Würzburg, Germany
{christopher.metter|wamser|trangia}@informatik.uni-wuerzburg.de

Abstract. Stateful migration processes for Cloud Services require the knowledge about their influencing parameters for the migration decision. Previous work focuses on the placement after the migration but not the migration process. In this work we evaluate the impact of network parameters on the migration performance as well as on the migrated applications. Therefore we propose an automatically set up testbed using OpenStack to measure key characteristics of the migration process.

1 Introduction

Cloud services are growing at a rapid pace and are expected to grow at an annual rate of 18 % in 2017 in a 246.8 billion dollar market [1]. The reason for this trend is based on the promise of almost unlimited and scalable resources the cloud provides. Typically, cloud services are scaled and distributed in the cloud to meet demand and service requirements. This is directly supported by the cloud infrastructure by providing capabilities for orchestration, placement, and migration of cloud services in the virtual environments so that resources can be released or used upon request. *Service Migration*, the process of moving a service from one physical host to another, in particular, is an integral feature of the cloud to allocate resources at another host or location, and to adapt for the services accordingly. The cloud thus meets its economic expectations, as costs can be controlled and requirements of the services can be met if necessary.

There are many different migration types. For stateless services, migration is simply done by booting up a second instance of the service on the target host, switching the endpoint to the new instance and then shutting down the instance on the first host [2]. For stateful services migration is more complex and multiple approaches exist tailored to different requirements on factors such as migration speed and performance during migration. While these migrations provide many possibilities, many cloud and service providers fear the use of stateful migrations due to missing ways to predict their effect during the migration as well as the migration duration.

Many works present approaches on the topic where to place virtual machines respectively where to migrate these based on different performance characteris-

tics [3–6] or on modelling the performance of virtualized systems [7–9]. Other work proposes models for VM migration times [10,11], propose a migration policy [12] or a migration progress management system [13]. Whilst these papers all introduce significant parts to plan and manage migrations, none provides detailed information on the used measurement methodology for the various migration phases and a detailed analysis of those measurements.

In this work we propose a testbed to analyze the impact of various parameters on the performance of migrations as well as on the performance effect of the migration on the migrated service. The testbed allows to measure the total migration time as well the migration phases. Using this testbed we benchmark migration processes under different network conditions (bandwidth, latency, drop rate) performing multiple migrations. Then we perform migrations of multiple applications under different condition simulating the migration inside a data center as well as the migration between data centers.

The contribution of this paper is

- the proposition of a testbed for migration benchmarking
- measurement and evaluation of the impact various network factors pose on migration duration
- measurement and evaluation of the impact the migration has on the migrated service under different network conditions

The remainder of the paper is structured as follows. In Section 2, related work is summarized and discussed. In Section 3, the background on cloud service migrations modes is outlined. The testbed is described in Section 4 whereas in Sections 5 and 6, the evaluation is presented and results are discussed. Conclusions are given in Section 7.

2 Related Work

This section features work and research with the focus on the performance analysis of virtual machine migration within data centers.

In [14] and [15] overviews and reviews on the common techniques and open research questions of virtual machine live migration are given. While the former focuses on giving a comprehensive literature research and summing it up, the latter provides a comprehensive survey on VM migration schemes. After introducing aspects of migration, state-of-the-art live and non-live migration techniques are reviewed and investigated. The authors conclude by summarizing open research questions that require solving in order to optimize VM migration.

A model predicting the duration of virtual machine migration is presented in [10]. At first, the parameters influencing the migration performance, and their dependencies, are identified. With the aid of two simulation models, predictions with an accuracy within 90% are possible. This paper gives only limited information on how the results were measured and which environment was used. Liu *et. al* propose a model to estimate the costs in terms of performance and energy consumption in [11]. Their approach is comparable to the previous work

and analyzes the key parameters that impact virtual machine migrations. Their model is evaluated using workloads in a Xen virtualized environment, presenting results with higher than 90% prediction accuracy.

According to [12] current migration algorithms based on a single objective lack the consideration of factors influencing the migration process. Therefore, they propose a migration policy that considers the migration process as a multi-objective problem. Testing their policy using CloudSim, their results promise an increased system performance.

[13] addresses the issue that currently no state-of-the-art live migration progress management system exists. In the opinion of the authors multiple problems arise from this lack of management. For example, it is possible that the performance of application, which is distributed over multiple machines, is degraded, as a split with increased delay between these virtual machines, leading to a higher latency, could occur. Pacer, their approach to a migration progress management system, addresses these issues by relying on run-time measurements of various metrics, analytic models and on-the-fly adaptation. Their experiments on a local testbed and on Amazon EC2 promise a high efficiency. The problem of disimproved dependencies among virtual machines after migration is also raised by [6]. AppAware, their contribution on this topic, evaluates dependencies between guests, and the placement of them on the hardware hosts in order to optimize the migration process. Simulations show that their proposal can lead to a decrease of the network traffic by up to 81% in comparison to non-application-aware technique.

In contrast to the presented work, in this work we conduct measurements with the OpenStack platform and analyze the performance factors in regard to the overall migration duration and its parts.

3 Migration Modes

This section provides the essential background information on how the different cloud migration modes work. A common and crucial feature all cloud environments support is the migration of machines. This describes the process of moving virtual hosts and/or services from one physical host to another. This technique is required for multiple reasons. At first, if a host running multiple virtual machines requires maintenance, e.g. due to broken hardware or a scheduled software update, the host needs to be shut down or rebooted. Other use cases are the dynamic resource management for load or power balancing within a data center, and to seamlessly migrate virtual machines from a test environment into production. The simple solution, to just deactivate the host, and therefore also its running guests, is not feasible, especially on a commercial platform where customers pay for the availability of their product and service level objectives have to be met. Therefore, techniques enabling the movement of guests from one host to another, minimizing the downtime for the customer, are required. In the following, the most common migration types are introduced.

Non-live migration This migration type is also known as cold or offline migration. At first, the availability of the required resources at the target host is validated. If enough resources are available, they are reserved on the target system. Now, the guest is shut down, then the virtual network of the guest is detached from the host, and the disk of the guest is moved to the target host. After completing this transfer, the virtual network connection is reattached at the target host and the guest system is restarted. This form of migration is noticed by the guest as a reboot.

Live Migration The next approach is the so called Live Migration. Its goal is to migrate a machine without disconnecting client and application. Memory, storage and network connectivity are transferred between the hosts. Instead of rebooting, the machine is paused on the source host, the image is moved to the target host and there resumed. There are several approaches to live migration and to increase the migration speed and thereby reduce the service's downtime.

Live Migration via Central Storage After the reservation of the resources on the target host a snapshot of the virtual machines memory is created and transferred while the machine is still operating on the source host. A shared central storage is mandatory for this type live migration. The guest machine is paused on the source host, its network connections are detached and the machine memory and its register content is transferred. Afterwards, the machine is resumed on the target host and its network connections are reattached. Depending on the transfer time of the remaining delta, the guest might only notice a sudden jump of the system time before and after the pausing of the machine.

Block Live Migration The second approach is called block live migration. Block migration is a similar process to the before mentioned live migration. This time, no shared storage is required as the disk(s) of the guest are located on the compute hosts and therefore are also migrated. Thus, the total the data volume rises in comparison to the live migration.

Pre-Copy Migration There are two ways to handle live migration. The first one is pre-copy migration. Since the machine is continuing its operation, the guest system and its memory most likely change during this transmission. Accordingly, the memory of the guest is compared to the already transferred content on the target host. If this delta is beyond a configured threshold, a new snapshot is created and transferred to the destination node. This process is repeated until the delta falls under a preset threshold. Then the machine is paused and the remaining delta is transferred. After resumption and reestablishment of the network connection the machine is fully operational without performance impact. The pre-copy migration can fail to ever complete if the threshold is set too low and/or the machine content changes too rapidly.

Post-Copy Migration The secord way is the post-copy migration. Here the process starts with suspending the machine at the source host and transfer a minimum of information (at least the register content) to the target host. The machine is then resumed at the target host and network is reattached. Then the

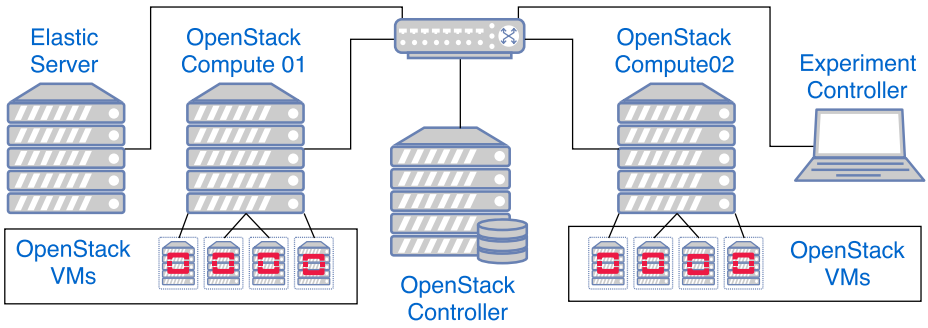

Fig. 1. Testbed Setup

remainder of the machine is transferred. During this phase page faults for not yet transferred pages are resolved over the network. The process is completed once the last fragments of the machine are transmitted. This process tackles the problem, pre-copy migration has with too rapidly changing content. On the other hand resolving page faults over the network can cause major performance problems.

Nomenclature: For nomenclature we orient ourselves at the OpenStack nomenclature, where *live migration* usually means *pre-copy live migration via central storage* and *block migration* is equivalent to *pre-copy block live migration*. These are also the two types we focus on in this paper.

4 Testbed Setup

Figure 1 presents the testbed used for the measurements presented in Section 5. A minimal OpenStack setup [1], sufficient to create virtual machines with network access as well as perform block and live migrations has been installed. It consists of the two compute nodes 01 and 02 (SunFire X4150, Intel Xeon 5300, 16 GB RAM, 500 GB HDD) running the virtual machines, and the controller node (Fujitsu Esprimo C5730 E-Star 5.0, Intel Core2 Duo E8400, 4 GB RAM, 250 GB HDD) also running the storage service for the live migration. The experiment controller is used to configure, start, and stop the measurement runs. For result recording an installation of Elastic Stack is used. Experiment results generated from OpenStack log files are collected via LogStash and forwarded to an ElasticSearch server. All of these hardware devices are interconnected via an 1000 Mbit/s Ethernet switch.

Templates for virtual machines are called flavors in OpenStack. These flavors allow the user to define the amount of virtual CPUs, the memory, the disk size, and the network connection of the guest, just as when purchasing a hardware server. For the upcoming measurements we used four out of the five default OpenStack flavors: m1.tiny, m1.small, m1.medium m1.large. Their according

[1] `https://www.openstack.org/`

Table 1. The used OpenStack Flavors and their properties

Flavor	VCPUs	RAM	HDD
m1.tiny	1	512 MB	1 GB
m1.small	1	2048 MB	20 GB
m1.medium	2	4096 MB	40 GB
m1.large	4	8192 MB	80 GB

properties are denoted in Table 1. Larger flavors where not explored due to memory requirements exceeding our testbed system.

5 Impact of Network Characteristics on the Migration Performance

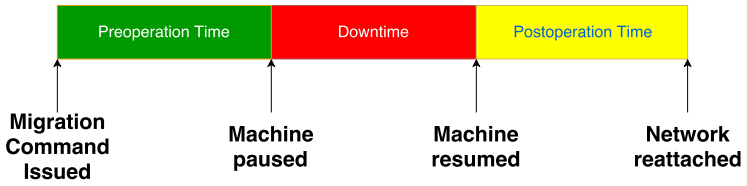

Fig. 2. Measurement Parameters

Figure 2 depicts the different stages of a migration that have been measured and evaluated. The migration time is composed of three blocks: *Preoperation Time*, *Downtime*, and *Postoperation Time*. The beginning of each migration is the execution of the migration command of a still running virtual machine. Now certain preoperations are executed until the machine is paused. This time is the *Preoperation Time*. Afterwards, the machines network is detached. The time until the machine is resumed on the target host is called the *Downtime*. Now the *Postoperation time* is running until the machines network is reattached. The relevance and portion of each step depends on the requirements of the migrated machine, or, accordingly, its service.

In the following the results of the measurements taken for network characteristic influence are presented and discussed. Network settings are modified and the above specified parameters are measured in the scenarios to be presented.

5.1 Impact of Network Throughput Limitations

A huge part of the migration time is allocated to the transfer of large amounts of data across the network. Therfore, migrations have been measured with bandwidth restriction of 1000 Mbit/s, 100 Mbit/s and 10 Mbit/s for the m1.tiny, m1.small, m1.medium and m1.large flavors.

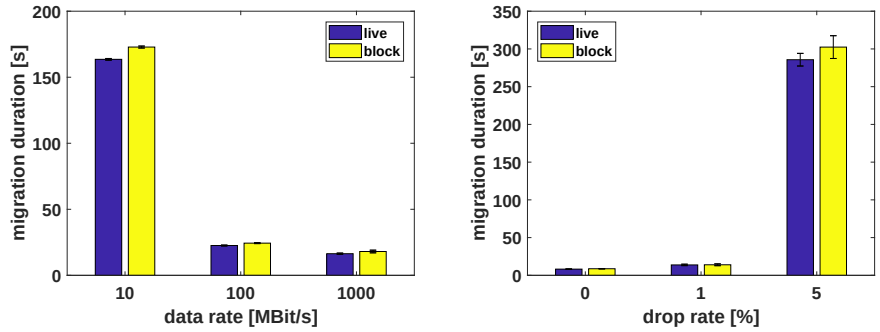

Fig. 3. Migrations of the medium flavor using different throughput limits (left) and network drop rates (right)

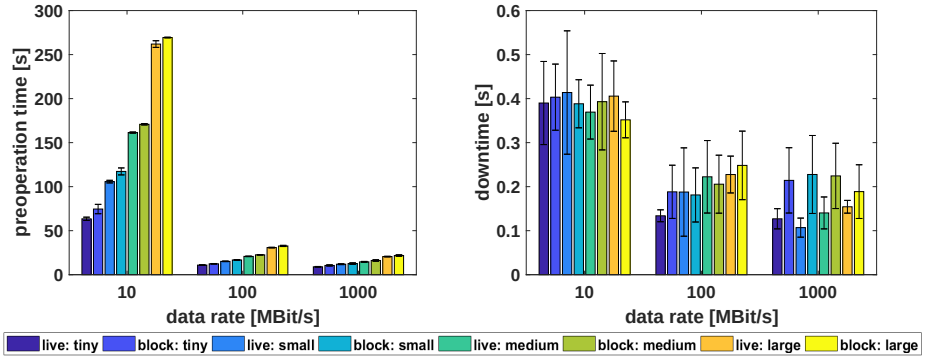

Fig. 4. Preoperation (left) and downtimes (right) for various flavors and migration types grouped by throughput

Figure 3 shows the migration times for the medium flavor. The migration time increases when the throughput is reduced. These results are representative for the other flavors. Between 100 Mbit/s and 1000 Mbit/s the difference is quite small compared to the difference between 100 Mbit/s and 10 Mbit/s. The average migration times of the live migration approach are lower than for the block migration approach.

The majority of the total duration is spent on the *preoperation time*. Therefore, it is also considerably affected by the throughput limit which relates to the copying of the first snapshot being located in the preoperation phase. The preoperation time is largely increased when reducing the available bandwidth. Live migration is faster than block migration. The downtimes are slightly sensible to the change of the throughput limits but the effect is weaker than for preoperation times since only a small part of the downtime phase relies on network transport. Live migration downtimes are always below the block migration downtimes by up to 50%. For the effect of the throughput limits on the *postoperation times* no statistical significant effect of neither the throughput limitation nor the migration mode can be found. The figure is therefore omitted.

5.2 Packet Loss in the Network and Related Effects

Another network parameter is the percentage of packets that are not successfully transferred e.g. due to uncorrectable bit errors in the line. This percentage is called the packet drop rate. The effect of the drop rate (d) depends on the algorithms used by OpenStack's migration mechanism and the transport protocol [16].

OpenStack uses different approaches in different modules making use of both UDP or TCP for the migration as well as normal operation. Measuring the effect of different loss rates is required to estimate the impact of unstable links on the different subtasks of the migration duration. Thus, migrations have been measured without error as well as for one and five percent drop rate. The five percent setting has been chosen as the upper bound for realistic scenarios, representing an really unstable data link (e.g. for a compute center this could be a cellular fallback). In an ideal scenario with ideal protocols, the respective minimal increase factor in transfer duration would be 1.0101 (one percent) and 1.0526 (five percent) according to Formula 1.

$$n = \sum_{i=0}^{\infty} d^i \tag{1}$$

Figure 3 exemplarily shows the migration times for the medium flavor with different drop rates. The increase between no and one percent drop rate varies between 38% (m1.tiny) and 77% (m1.large) with block migration. The increase between one and five percent drop rate is even larger. The duration rises between 1164% (m1.tiny) and 2525% (m1.large). Any of these factors significantly exceeds the theoretical optimal factors that have been calculated above. This leads to the conclusion that a lot of packets are redundantly transferred and the link does not operate near optimal speed.

Independently of the drop rate, the live migration is slightly faster than block migration. While the relative advantage is constant, the absolute advantage increases with instance flavor and drop rate culminating at almost 23 seconds for the large flavor at five percent drop rate.

Again the *preoperation time* is the largest part of the total migration duration as depicted in Figure 5. Similar to the duration, the live migration has a slight advantage over the block migration. The factor is even higher than for the total duration, leading to the assumption that the part influenced most by the packet loss is the transmission of the snapshot for the block migration and the synchronization of the network file system for the live migration. Ignoring singular exceptions, the average *downtime* increases slowly with the drop rate. For the first step only few scenarios (e.g. block with m1.medium) reach and exceed the factor of two. A slightly larger increase is visible at the second step with all flavors and modes at least doubling. On average live migration has a slightly better downtime but the confidence interval overlap. The behavior of the *postoperation time* is less sensitive to the increased drop rate. Figure 5 also shows the postoperation times. There is a slight increase between no and one

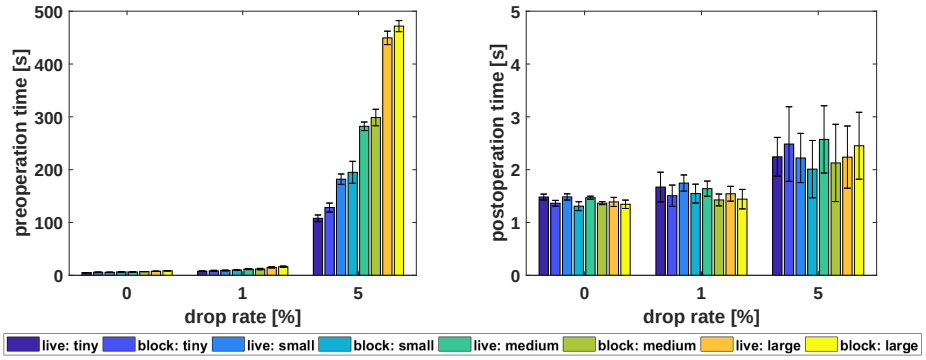

Fig. 5. Preoperation (left) and Postoperation times (right) for various flavors and migration types grouped by drop rate

percent drop rate and a slightly larger increase when increasing the rate to five percent. The increase is relatively small compared to the preoperation time at about sixty percent between zero and five percent drop rate. As for downtime, there is no significant difference between live and block migration.

6 Evaluation of Migration Performance from Application Perspective

Applications have different requirements regarding the migration process. There are less time critical applications like downloads of large files. However, there are also more time critical applications with different requirements. Normal video streaming requires an acceptable quality with no stalling while live streaming requires a short transmission time to the user as well. The usability of some applications during migration has been tested to evaluate the effect of the migration on the application and ultimately the user's quality of experience (QoE) when using the application.

Three different approaches to video streaming were tested in two scenarios. *Scenario I* features no extra latency and a throughput of 1000 Mbit/s, which is similar to the environment parameters inside a data center. The migration at 100 Mbit/s and 100 ms delay in *Scenario II* resembles the migration over a mediocre link between remote data centers. For the video content the Sintel[2] movie has been chosen due to its licensing and its popularity. An application scenario is considered migrate-able, when the migration successfully completes and no noticeable impact for the user is visible.

6.1 Server-based Streaming Using Colocated Content

Live video streaming requires the parallel encoding and distribution of the material. Performing both tasks on a single machine puts high requirements on

[2] https://durian.blender.org/download/

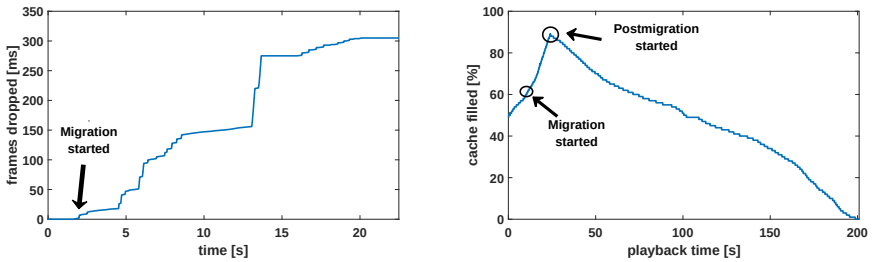

Fig. 6. Left: Frames dropped during the beginning of the migration process using server side streaming with co-located content in *Scenario I* **Right:** Percentage of buffer fill level over time before, during and after migration using server side streaming with external content in *Scenario I*

the CPU to achieve the encoding in real time. For this task, the widely utilized ffmpeg[3] was used. For the actual encoding, the ffmpeg application was used to convert the source video into a buffer file. For the streaming itself, ffserver was used to transfer the aforementioned buffer file. In this scenario, the streaming intelligence is supposed to be on the server side. Therefore, MPplayer[4] has been chosen for this task. MPlayer does neither automatically reconnect nor adapt the streaming quality. It just plays a video from a provided URL.

A server of the m1.large flavor is setup to evaluate the migrate-ability of this scenario. The large flavor is necessary to provide enough computing power to run the encoding since it provides four vCPUs.

After the migration is triggered in *Scenario I* some transmission errors occur when no cache is enabled. Leading to a large number of dropped frames as seen in Figure 6. To the user this is visible as a stuttering playback of the stream and sudden jumps of a few seconds ahead. If sufficient caching is enabled, this problem is not visible to the user. Only the cache level drops slightly. After most of the machine is migrated, the server migration is taking a lot of time for the remaining part. The log file shows that the content remaining to be transfered permanently increases. This is due to the fact that encoder permanently encodes the video and writes to the buffer file. This leads to a never-ending migration process. It is permanently stuck, alternating between zero and five percent of remaining content. Even after 20 minutes the migration process has not finished. As soon as the encoding is terminated remotely, it is only a matter of a few seconds and the migration finishes. The experiment has been repeated with *Scenario II*. This time, it takes longer to reach this loop state. Additionally, the video playback also stalls. The fact that the migration not only noticeably impairs playback quality but also never completes renders this application scenario *not migrate-able*.

[3] https://www.ffmpeg.org/
[4] http://www.mplayerhq.hu/

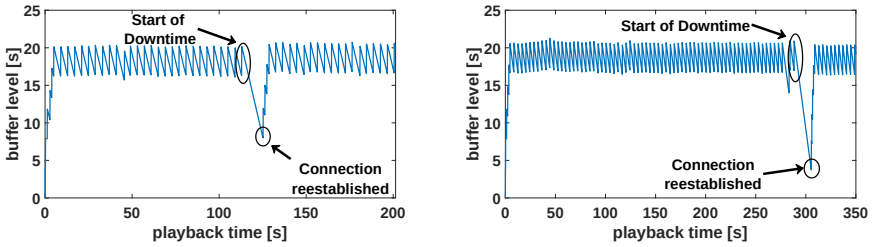

Fig. 7. Percentage of buffer fill level over time before, during and after migration using DASH in *Scenario I* (left) and *Scenario II* (right)

6.2 Server-based Streaming of External Content

As the concept detailed in Subsection 6.1 fails, the idea is to move the encoding to a separate node and migrate only the node running the streaming server. The used applications remain the same. With the required amount of processing power significantly reduced, it is possible to switch to the smaller m1.small flavor.

In *Scenario I* the migration is always performed - meaning the virtual machine is moved to the other server and the ffserver process continues its operation. Unfortunately, in 48% of the migration runs the client disconnects during the postoperation phase. If the cache is enabled, the playback continues until cache is depleted, as seen in Figure 6. If no cache is enabled, the playback instantly terminates. It is required to restart the player to resume playback. Therefore, this scenario is only *semi-migrate-able*, as a high chance of failure renders it inapplicable for production usage where the end-user should, in best case, not recognize the migration.

A very interesting behavior occurs when migrating in *Scenario II*. One might usually expect that under worse conditions the QoE would further decrease during the migration. This is not the case. Total migration takes longer especially due to the increased latency and is successfully completed as in *Scenario I*. The connection stays stable with the cache not filling for a short amount of time that correlates with the expected lengths of downtime and postoperation time. However, if no cache is enabled, the video stalls for a long period and then continues where it was suspended. This is a very surprising behavior. The only possible explanation is that the disconnect is not detected fast enough due to the already existing network delays. In this scenario the migration is *fully migrate-able*.

6.3 Dynamic Adaptive Streaming over HTTP (DASH)

As a final scenario for video streaming, a client-based streaming application was chosen. On the server side the material was provided in multiple quality levels by a web server and a playlist file was provided to the client. No further intelligence or optimization happened on the server side. The TAPAS palyer [17] (Tool for rApid Prototyping of Adaptive Streaming algorithms) was chosen on the client side. It is capable of adaptively streaming playback.

Application	Scenario 1	Scenario 2
Download	✓	✓
Server Side Streaming + Content	✗	✗
Server Side Streaming w/o Content	✗*	✓
Client Side Streaming	✓	✓
Video Gaming	✗*	✗

Table 2. Summary of the migratability of different applications. Where ✓means the migration is possible and works reliable and ✗means that it does not. ✗* means that the migration is successful some times but fails at others or causes non tolerable impairments. Scenario 1 resembles intra compute center migrations while Scenario 2 resembles inter compute center migrations.

In *Scenario I* the migration is successfully performed. The server application survives the migration and the client successfully reconnects and continues streaming the video. During the time where no network operation is possible the clients cache level drops. With a preset cache level of 20 seconds the migration is possible with fluid playback of the video stream as shown in Figure 7. In the tested configuration after the first segment of the video the player constantly operates at the maximum quality possible even during the migration.

The experiment was repeated for *Scenario II*. The first obvious change was the far prolonged migration duration. The migration was again performed successfully without any negative effects on the server application. Again, the client reconnected successfully. This time the buffer level dropped farther, as seen in Figure 7. Also, the application adapted due to the low buffer level, requesting the first segment after the reestablishment of the link in a lower quality level.

The migrations for the client side streaming application were always successful with no failures or total loss of connection, as observed for the server side applications. This renders this solution *completely migrate-able* in both scenarios.

6.4 Other Applications and Summary

We have also evaluated the migrate-ability of further applications. Migrating a simple file server in a m1.small flavor has been successful without any problems. Since this is a simpler version of the client side streaming this is little surprise.

We also have tested whether migrating a game server during a running game was possible. Therefore we ran a dedicated *Counter-Strike: Source* in a m1.medium flavor. The migration always succeeded but during the migration the game was unplayable. The preoperation phase caused major stuttering and lags. The downtime caused a longer lag while during the postoperation phase the AI players continued to operate while the human players were still waiting to reconnect. Thus we designate this scenario as not migrate-able.

An overview of the migrate-ability can be found in Table 2.

7 Conclusion

Due to the increasing popularity of cloud services, resource management with respect to the users' perceived quality, as well as in terms of energy efficiency and cost, is becoming more and more important in a cloud infrastructure. A fundamental part within this resource management process in the cloud is the migration of cloud services. The decision whether the benefits of a migration justify the migration effort requires addition information and studies. This includes in particular the influence of the migration on the actual service performance, the duration of the migration, and influences of different network parameters on these factors.

In this paper an overview of the related work has been given and the background is presented. A testbed to measure migrations has been described. The effect of the network parameters throughput and drop rate has benn analyzed, showing that both parameters have different effects on the three phases of a migration.

Next, the migration performance was analyzed from application perspective. To assess this performance multiple applications have been chosen and migrated inside the testbed while clients were connected and using the provided services. The assessed applications include server-based video streaming with and without content inside the virtual machine and adaptive streaming using DASH. While the applications with little server side intelligence had few problems with the migrations the more complex applications failed to migrate at all or suffered severe impairments to the usability. More precisely the server side video streaming with included content did not migrate at all while on the other hand, the DASH streaming migrated without problems.

Future work may deal with developing concepts to make migrations application-aware. Thus, allowing the migration process to adapt to the used application and improve the quality during migration resp. making migration possible at all. Additional measurements to decrease the granularity are to be taken and further parameters (e.g. I/O load) are to be evaluated. With enough additional data, a model for migration duration should be designed.

Acknowledgment

This work was partialy supported by German Research Foundation (DFG) under Grant No. KO 3445/11-1. and the H2020 INPUT (Call H2020-ICT-2014-1, Grant No. 644672).

References

1. Pettey, C., Goasduff, L.: Gartner Says Worldwide Public Cloud Services Market to Grow 18 Percent in 2017 (February 2017)

2. Fehling, C., Leymann, F., Ruehl, S.T., Rudek, M., Verclas, S.: Service migration patterns–decision support and best practices for the migration of existing service-based applications to cloud environments. In: Service-Oriented Computing and Applications (SOCA), 2013 IEEE 6th International Conference on, IEEE (2013) 9–16
3. Piao, J.T., Yan, J.: A network-aware virtual machine placement and migration approach in cloud computing. In: Grid and Cooperative Computing (GCC), 2010 9th International Conference on, IEEE (2010) 87–92
4. Mohammadi, E., Karimi, M., Heikalabad, S.R.: A novel virtual machine placement in cloud computing. Australian Journal of Basic and Applied Sciences **5**(10) (2011) 1549–1555
5. Hyser, C., McKee, B., Gardner, R., Watson, B.J.: Autonomic virtual machine placement in the data center. Hewlett Packard Laboratories, Tech. Rep. HPL-2007-189 **189** (2007)
6. Shrivastava, V., Zerfos, P., Lee, K.W., Jamjoom, H., Liu, Y.H., Banerjee, S.: Application-aware virtual machine migration in data centers. In: INFOCOM, 2011 Proceedings IEEE, IEEE (2011) 66–70
7. Huber, N.M.: Autonomic Performance-Aware Resource Management in Dynamic IT Service Infrastructures. PhD thesis, Karlsruhe, Karlsruher Institut für Technologie (KIT), Diss., 2014 (2014)
8. Noorshams, Q., Busch, A., Rentschler, A., Bruhn, D., Kounev, S., Tuma, P., Reussner, R.: Automated Modeling of I/O Performance and Interference Effects in Virtualized Storage Systems. In: Distributed Computing Systems Workshops (ICDCSW), 2014 IEEE 34th International Conference on, IEEE (2014) 88–93
9. Vaupel, R., Noorshams, Q., Kounev, S., Reussner, R.: Using Queuing Models for Large System Migration Scenarios–An Industrial Case Study with IBM System z. In: Computer Performance Engineering. Springer (2013) 263–275
10. Akoush, S., Sohan, R., Rice, A., Moore, A.W., Hopper, A.: Predicting the performance of virtual machine migration. In: Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on, IEEE (2010) 37–46
11. Liu, H., Jin, H., Xu, C.Z., Liao, X.: Performance and energy modeling for live migration of virtual machines. Cluster computing **16**(2) (2013) 249–264
12. Sallam, A., Li, K.: A multi-objective virtual machine migration policy in cloud systems. The Computer Journal **57**(2) (2014) 195–204
13. Zheng, J., Ng, T.E., Sripanidkulchai, K., Liu, Z.: Pacer: A progress management system for live virtual machine migration in cloud computing. IEEE transactions on network and service management **10**(4) (2013) 369–382
14. Kapil, D., Pilli, E.S., Joshi, R.C.: Live virtual machine migration techniques: Survey and research challenges. In: Advance Computing Conference (IACC), 2013 IEEE 3rd International, IEEE (2013) 963–969
15. Ahmad, R.W., Gani, A., Hamid, S.H.A., Shiraz, M., Xia, F., Madani, S.A.: Virtual machine migration in cloud data centers: a review, taxonomy, and open research issues. The Journal of Supercomputing **71**(7) (2015) 2473–2515
16. Mathis, M., Mahdavi, J., Floyd, S., Romanow, A.: TCP Selective Acknowledgment Options. RFC 2018 (Proposed Standard) (October 1996)
17. De Cicco, L., Caldaralo, V., Palmisano, V., Mascolo, S.: TAPAS: a Tool for rApid Prototyping of Adaptive Streaming algorithms. In: Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming, ACM (2014) 1–6