International Journal of Innovative Computing, Information and Control
Volume **11**, Number **3**, June **2015**

ICIC International ©2015 ISSN 1349-4198
pp. 973–983

LIVE MIGRATION OF VIRTUAL DISTRIBUTED COMPUTING SYSTEMS

Jan Fesl[1,2], Luboš Plch[1], Marie Doležalová[1] and František Drdák[1]

[1]Institute of Applied Informatics
University of South Bohemia
Branišovská 31, České Budějovice, 370 01, Czech Republic
fesl@post.cz

[2]Department of Computer Science
Czech Technical University in Prague
Karlovo Náměstí 13, Prague, 121 35, Czech Republic

Received April 2014; revised November 2014

Abstract. *The live migration technique is a very hot topic today in connection with the virtualization technology, which is widely used in different computing environments from the single processor computers to the large cloud solutions and data centers at present. Live migration of the given virtual machines brings some aspects, which can cause troubles in comparison to much easier offline migration of them. We have studied current architectures used for live migration and their optimizations for more efficient migration between various physical hosts. The specific part of our research involves studies regarding the live migration of the group of virtual machines representing either cooperating cluster nodes or distributed computing system. From the point of view of their complete live migration they have similar behaviour. The basic aspects and principles of this type of migration functionality are discussed in the following sections. This paper concludes by the introduction of new live migration algorithm, which can be suitable for automated live migration of various distributed architectures as a whole.*
Keywords: Live migration, Virtualization, Memory compression, Distributed computing, Cluster, Grid

1. Live Migration of Virtual Machines.

1.1. **Virtual machine components.** The virtual machine is a concept, which divides one physical computer into many virtual computers. Computing resources of this physical computer are shared among all virtual machines. There are two basic concepts used for virtualization [1] – hardware and software (sometimes called paravirtualization). The first type is more powerful, but needs one physical computer dedicated to virtualization, and the second is largely based on some part of operating system and allows larger system utilization for various applications. Every virtual computer consists of three main parts – virtual (emulated) computer hardware, virtual memory and virtual data storage (hard disk). Virtual data storage is mostly located on shared network area storage device with a network file system which simply allows its utilization by individual physical hosts.

1.2. **Virtual machine migration – strategies and principles, current state.** There are two main strategies of a virtual machine migration – offline and online. First type is much simpler, because the whole migration process consists only of three main steps. Following diagram shows the migration of a virtual machine from physical host A to another physical host B. More detailed information can be found in [2].

Three phases of an offline migration process are as follows.

1) Virtual machine (VM) pushing and stopping on physical host A.
2) Copying saved memory (M) and virtual hardware settings (H) of VM to physical host B. Data storage (hard disk) at B is mostly only reloaded from a shared network device.
3) Execution of VM (M, H) on physical host B.

This technology is today well known and widely used for virtual machine migration. The main disadvantage is that some services provided by virtual machine are not available during the migration process (lasting tens of seconds or more [3]). This solution cannot be readily utilized for services such as DNS, SMTP, LDAP, RADIUS because these services require high availability. Similar situation appears in distributed computing systems, because it is not really possible to stop all the nodes at the same time.

A live virtual machine migration is a more complicated problem, because memory content of the migrating machine is under permanent change all the time. There is another limitation caused by different available data throughput into virtual machine memory in the scope of the same physical environment and between different physical hosts connected by means of the network. A maximal theoretical data throughput to the memory of a virtual machine can be hundreds of gigabits per second [4] unlike throughput between two networkconnected physical hosts, where it may be only tens of gigabits per second – a typical value for infiniband, latest ethernet technology and fiber channel. Frequent changes in memory pages and low network throughput can cause impossibility to finish the migration process.

Live memory migration scheme is described in detail in [2,3]. The basic migration scheme is as follows. A virtual machine VM is migrated from physical host A to physical host B. Gustaffson's work [22] tries to accelerate this phase using the ultrafast network area storage.

There are six phases of live migration.

1) Pre-Migration. An active VM runs on physical host A. To speed up any future migration, a target host may be preselected where the resources required to receive migration will be guaranteed.
2) Reservation. A request is issued to migrate given VM from host A to host B. B must confirm that it has the necessary resources and reserves a VM container of appropriate size. Failure of resource reservation here means that the VM simply continues to run on unaffected.
3) Iterative Pre-Copy. During the first iteration, all memory pages are transferred from A to B. Subsequent iterations copy only those pages modified during the previous transfer phase.
4) Stop-and-Copy. Suspend the running VM instance at A and redirect its network traffic to B. CPU state and any remaining inconsistent memory pages are then transferred. At the end of this stage there is a consistent suspended copy of the VM at both A and B. The copy at A is still considered to be primary and is reactivated in case of failure.
5) Commitment. Host B indicates to A that it has successfully received a consistent VM image. Host A acknowledges this message as commitment of the migration transaction: host A may now discard the original VM, and host B becomes the primary host.
6) Activation. The migrated VM on B is now activated. Post-migration code runs to reattach device drivers to the new machine and reuses the IP addresses.

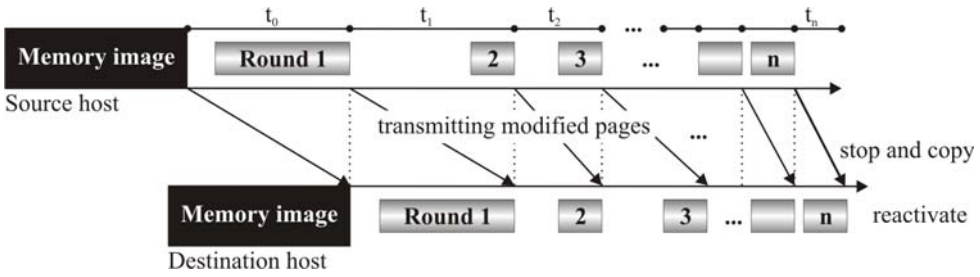

FIGURE 1. Live computer migration scheme

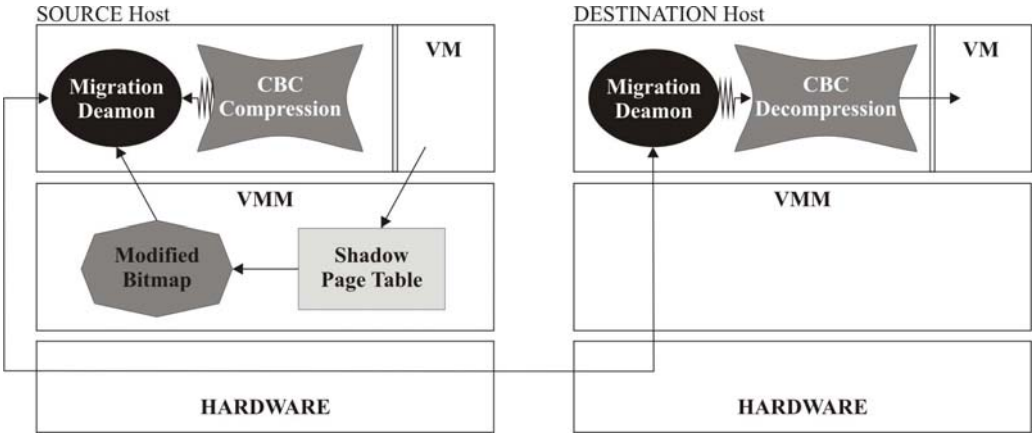

FIGURE 2. Compressed memory migration diagram

1.3. **Memory compression for faster migration.**

1.3.1. *Memory compression mechanism.* Memory compression is a technique, which allows radical decreasing the amount of necessary data to be transferred. This solution is described in detail in [5] and similar solution is actually used in Hyper-V Windows 2012 R2 [9] or VmWare Sphere 5.1 VMotion technology [10]. Solution described by Jin et al. [5] deals with the analysis of memory pages data granularity and the choice of the compression algorithm suitable for specific memory page. WKdm [6] and LZO [7] algorithms were selected and tested for memory compression. The memory compression scheme and migration process are depicted in Figure 2.

Additional work published by Gustafsson [8] discusses the use of the universal common compression tools such as gzip, which is not suitable for this solution. Another more advanced approach to memory compression used a high-sophisticated adaptive compression method [25].

1.3.2. *The benefits of memory compression.* Jin et al. [5] proved in practice that by using an efficient compression mechanism, live migration time can be reduced by about 30% and the amount of transmitted data can be reduced up to 60%. This time reduction contains also the necessary compression time. Gustafsson's work provides comparable results to Jin's.

2. **Virtual Computer Cluster and Distributed Computing Model.**

Distributed computing is the special science discipline dealing with processing complex computing problems, which cannot be processed by a single standalone computer (for example processing very large data sets). Message communication between cooperating nodes of the

distributed system is achieved mainly by message passing. State of a distributed computing system [11] is defined as a set of computer nodes with their own memory (containing computing processes) and set of messages, which have been sent but still have not been delivered. Messages from the communication channels directly affect the computing processes with changes to the memory pages of the appropriate computer node. For the live migration of given distributed system, it is necessary to guarantee also the delivery of such communication messages.

2.1. **Virtual distributed system architectures.** Virtual distributed system architectures are similar to the real distributed systems. Message passing between virtual communicating nodes is exactly the same as in the real distributed system. Messages (packets) from the virtual source nodes are processed by the virtual machine and passed to the operating system, which completes message transmission to another virtual member node. There are three basic concepts of distributed system architecture [11]. The first is the common client-server model, the second is peer to peer model (p2p) and the third is a combination of the client-server and the p2p model.

2.1.1. *Client server model live migration.* The migration of the client server model needs to ensure migration of one virtual server node and many virtual client nodes from their physical hosts to other physical hosts. This computing model is often used in many parallel software architectures [12] such as in mpi, open mpi, mpich, lam/mpi and various computing cluster solutions like MOSIX, openMosix, Kerrighed or OpenSSI. All previously mentioned architectures are able to be run in the virtualized environment and can be used for distributed computing. The live migration of a server node is much more difficult in comparison to a client node because the server node's memory content is modified by messages from all client nodes. Migration of the client node is easier because only one server node sends messages to this client meaning fewer messages modify client's memory content.

2.1.2. *Peer to Peer model live migration.* Peer to peer systems have the all individual computing nodes equivalent. An average count of communication messages should be equal on all nodes. A set of algorithms for group coordination exists in distributed computing, which also works with p2p topology, but needs a coordinator [13]. Typical examples of these algorithms are algorithms for entering into critical sections, leader election or replication. This fact means that some nodes can have a communication message count higher in comparison to others.

2.1.3. *Real solutions for virtual cluster/grid infrastructures.* For remote management of virtual machines, which are placed on various physical hosts and can create a virtual distributed system, the following tools such as Microsoft System Center (hypervisor Hyper-V) [14], VMware VCenter (VMware) [15], OpenNebula (Xen, Kvm, VMware) [16], OpenStack (KVM, LXC, QEMU, UML, VMware, Xen, Hyper-V, Bare Meta) [17] or OpenQrm [18] are commonly used. Some of them require using a specific hypervisor. More information can be found in [19]. The live migration of only one virtual machine is supported by Microsoft System Center, Vmware Vmotion technology and Citrix XenServer [20]. Till now none of these solutions supports live migration of clusters or distributed systems.

2.2. **Virtual distributed system live migration problems.** In the following sections we will discuss some problems which can have an impact on the migration process. Their elimination is important for a safe migration process and creates basics of a scheme, which will be introduced in the end of the article.

2.2.1. *Free resources of physical hosts involved in migration.* The virtual distributed computing system is usually represented by tens, hundreds or thousands of virtual machines. Complete migration of such given amount of virtual machines requires the successful allocation of all needed resources at the destination computing environment. This allocation is done during the pre-migration and reservation phase and must be done dynamically. Some aspects of the migration suitability will be discussed.

2.2.2. *Low throughput of the communication network.* Network throughput between the source and the destination physical node can be very low in comparison to the throughput requirements for needed virtual machine memory updates. This can cause live migration impossible, because the difference of memory content between the migrating and migrated machine stays the same, or even gets larger. This must be detected in the pre-migration phase. If it is not possible to allocate the necessary bandwidth, migration cannot be executed.

2.2.3. *Message loss during the migration phase.* The current global state of a virtual distributed system is defined as a set of virtual machine memory contents and contents of all communications channels (messages, which have been sent, but have not been delivered). If a message (packet) is lost during migration, the current global state is corrupted. Some self-repairing protocols are resistant to these types of errors, but live migration technique should not modify communication conditions and should be reliable and transparent. None of currently used hypervisors contains techniques for message loss correction.

2.2.4. *Network bandwidth saturation during the migration phase.* Given virtual distributed system is represented by a set of virtual nodes. Virtual nodes can be migrated in parallel or serially. Parallel migration is faster but still, a migration of a large number of virtual nodes can easily cause network bandwidth saturation (typically gigabytes of memory content per one node are migrated). This leads to a lack of availability of some services on the network and various other problems. More information can be found in [23].

3. **Live Distributed Architecture Migration Scheme.** A new migration scheme, which is suitable for a very large data centres, proposed in this paper consists of three main parts: network topology optimization, destination host selection, and finally the live migration process. In this proposal we assume that the migration process is managed only by one central coordinator.

3.1. **Network bandwidth optimization.** There are three types of network traffic in the virtual distributed system. The first is communication traffic (CT) among all the nodes of the distributed system typically produced by passing of messages. The second type of traffic is generated by the migration of virtual machine's memory content (VMCT) from one physical host to another. The third type is generated by reading/writing from/to virtual hard drive network area storage (NAST). CT and NAST are generated during the whole lifetime cycle of the virtual distributed system. VMCT is generated only during the migration process, but there is much more of it in comparison to CT or NAST. VMCT can cause bandwidth saturation and has negative influence on other traffic. The theoretical solution is to divide the network bandwidth into three sub-bandwidths. This division can be accomplished through the usage of three separate (virtual or physical) network circuits, where each circuit is dedicated only to a specific traffic type. Another solution consists of the sharing of one band, which can be divided and allocated through a particular attribute of service-providing protocols (such as RSVP [21]).

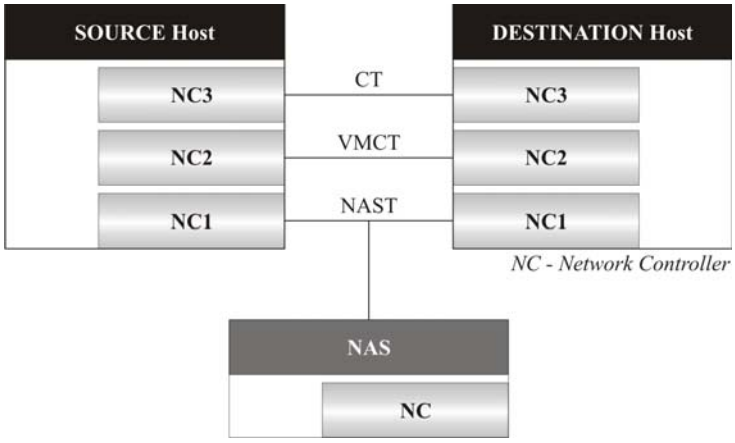

FIGURE 3. Three sub-bands of network bandwidth

3.2. **Selection of destination physical hosts.** A typical distributed system can contain hundreds or more of virtual computing nodes distributed among set of physical nodes. The determination of the new distribution of the set of virtual machines among different set of physical nodes is very complicated task. Typical reason for the reallocation of the set of virtual nodes is the need of releasing their concrete physical environment (changes in the hardware equipment, etc.). The main idea of the following algorithm is to solve this reallocation problem. The simplest solution is to migrate all given virtual machines of the given computing system from their current physical host to the new one at once, but this is usually not possible, since computing capacity of selected destination physical node is not sufficient.

A formal definition of our migration algorithm is as follows, the central coordinator C manages the migration process. The virtual distributed system uses message passing for communication purposes. We assume that the probability p, of message being sent between any two nodes is the same and is independent on their physical location. S_1 is a set of source physical nodes; S_2 is a set of destination physical nodes. DS (V, S_1, ID) is a distributed system with a set of virtual machines V, running on S_1 with a specific identification (ID). All virtual machines are the members of the same computing system. The ID is independent of the physical nodes.

S_1 may have same common items with S_2. S is a set of all physical nodes. $S = S_1 \cup S_2$.

$M(S_1, S_2)$ is an operation migration, which enables transfers of all virtual machines running on S_1 to S_2, $N(S_1)$ or $N(S_2)$ is the count of physical nodes contained in S_1 or S_2.

For a distributed system, let us make $M(S_1, S_2)$ with minimal $N(S_2)$. It is possible to achieve minimal $N(S_2)$ in the following three steps.

1) C sends a message A with an ID of the virtual distributed computing system to all the nodes of S_2 and obtains back a message B from each node. The message B contains the count of free resources. If at node X, which has sent back a message B to C, some virtual machines are running as a part of the distributed system with a specific ID, the message B contains the count of free resources incremented by the count of virtual machines with the same ID running on X.
2) C makes a list of resources from the incoming messages and sorts it in descending order according to the number of free resources – nodes with the highest amount of resources will be used first. Also list of requests of resources for the members of S_1 is prepared and sorted in the same order. Then the verification relative to the intended migration solution existence is performed from the point of view of the resource mapping possibility between the two lists.

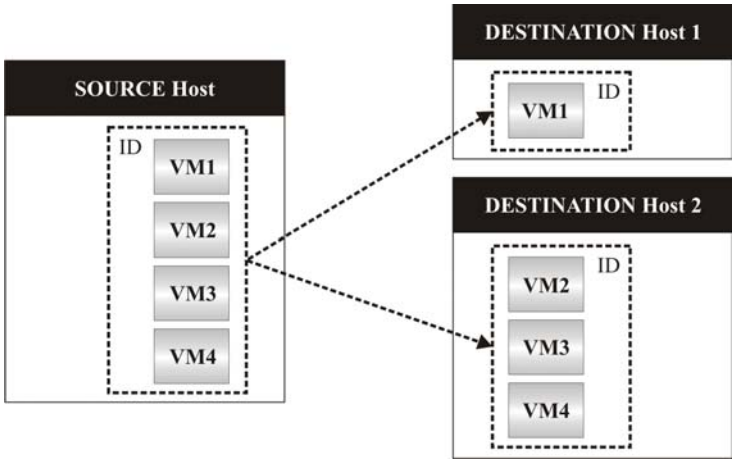

FIGURE 4. Destination nodes for migration discovery

3) If the solution exists it is produced by C recursive selections from the lists comparing the needed amount of resources until all items of S_1 are mapped onto S_2.

The algorithm finds (if existing) the nodecount optimal solution, which will be a part of the premigration and reservation phase. (If the solution does not exist the set S_2 must be modified.)

3.3. **Distributed virtual system live migration process.** In previous section we mentioned the resource allocation algorithm, which serves as the input for the next phase. $DS(V, S_1, ID)$ is the current distributed system, which is running on nodes from the set S_1 and $DS(V, S_2, ID)$ is a new distributed system, which will be relocated from S_1 to S_2.

3.3.1. *Virtual remote network message buffer.* During the Stop and Copy phase the current migrating virtual machine can obtain one or more communication messages from other nodes. The virtual buffer stores these messages into queue on a node from S_1 and delivers them to the similar buffer assigned to the node from S_2. The main function of this buffer is to save the messages, which could be lost during migration and to enable the delivery of these messages to the destination node.

3.3.2. *Algorithm for migration of whole distributed system.* If there is the need to migrate the whole distributed system part by part we can use following method. V is a set of all virtual machines of some virtual distributed system. S_1 is a set of source physical nodes; S_2 is a set of destination physical nodes.

1) Select k virtual machines from V, which are currently running on nodes from S_1, $0 < k < N(V)$, create a new empty set V_k, insert all selected virtual machines from V into V_k and remove these virtual machines from V.
2) For all items in V_k, start the pre-copy phase.
3) Before starting the migration process, create virtual network buffers on all nodes from S_1, where selected virtual machines are currently running. Message capturing on all nodes is disabled.
4) Create virtual network buffers on all corresponding nodes from S_2 for the virtual machines being migrated and interconnect this buffer to buffers created in stage 3.
5) Activate message capturing into virtual network buffers on all source nodes from S_1 and transmit this message to virtual network buffers at destination nodes from S_2.
6) For all items in V_k, start the post-copy phase.
7) For all nodes in V_k, start the commit phase.

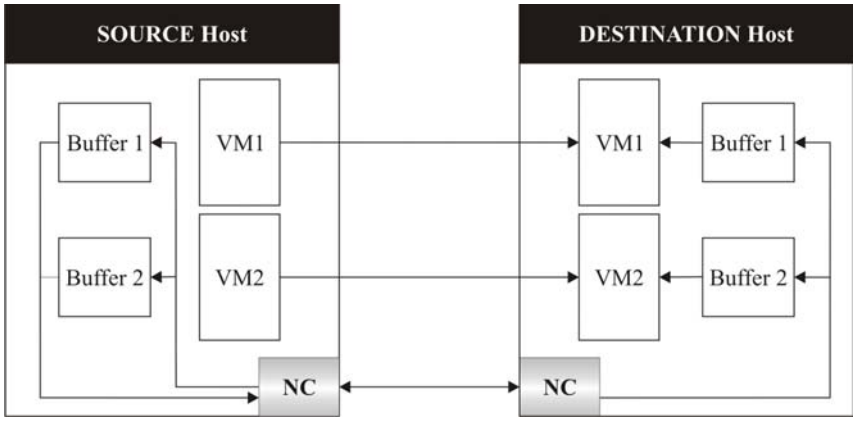

FIGURE 5. Distributed system migration algorithm

8) Start all virtual machines that have been migrated and deliver messages from their corresponding network message buffers.
9) Stop all virtual network buffers.
10) If V is empty, stop the migration process, else continue with the stage 1.

4. **Proposed Algorithm Simulation Results.** For the algorithm from previous section we had upgraded the standard open source Xen hypervisor. The real simulation was done via the university multiprocessor cluster, which consists of four computing nodes, one network area storage and one management node. All of these nodes were interconnected via the 10 Gb/s ethernet network. Every computing node contained two physical ten core Intel Xeon processors and 128 GB operating memory. On every physical node were started ten dual-core virtual nodes with Linux Debian operating system including the Xen (upgraded Xen) hypervisor.

4.1. **System transparency testing.** The first simulation was done for the verification that the migration process of the distributed system is really transparent. The simulation was as follows. The same computing process (tasks including communication over BSD sockets) on all virtual nodes was executed. All executed processes communicated via UDP communication protocol. UDP protocol was selected, because it contains no means for packets loss correction. The graph in Figure 6 shows, that our implementation significantly improved packet loss during the migration phase. We tested the migration process for the minimal and the maximal ethernet MTU size to eliminate the network media influence.

4.2. **System downtime measurement.** The next simulation depicted in Figure 7 shows the average downtime for the migrating virtual nodes during the migration phase. Our solution was in this case a bit slower in comparison to the standard Xen solution due to the overhead of the virtual network message buffers.

4.3. **Migration time of whole distributed system.** The time of migration of the distributed system, which was in this case measured, depends mainly on the migrating nodes count and their operating memory size. For the migration of the larger amount of virtual nodes is possible to achieve the network bandwidth saturation. The various colours of lines in the graph depicted in Figure 8 show various memory sizes of migrated virtual nodes.

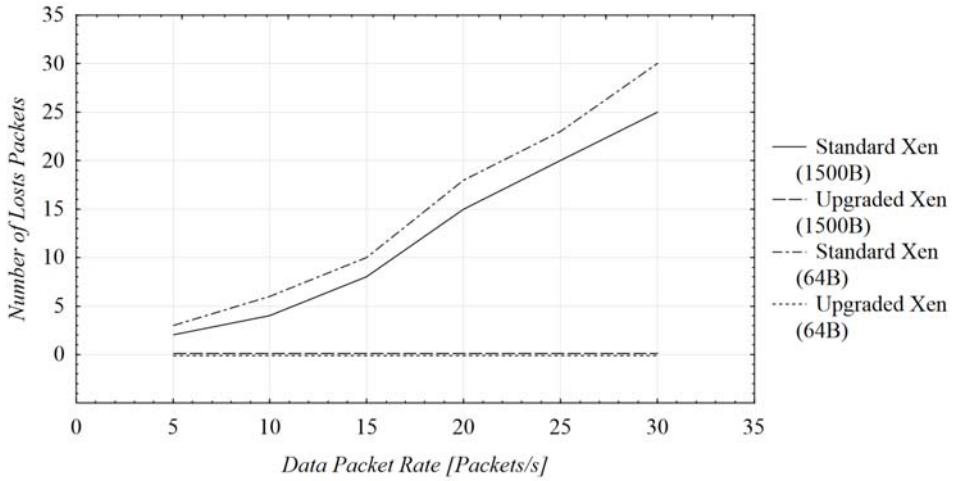

FIGURE 6. Message packet loss, measurements for 1500 and 64 bytes MTU

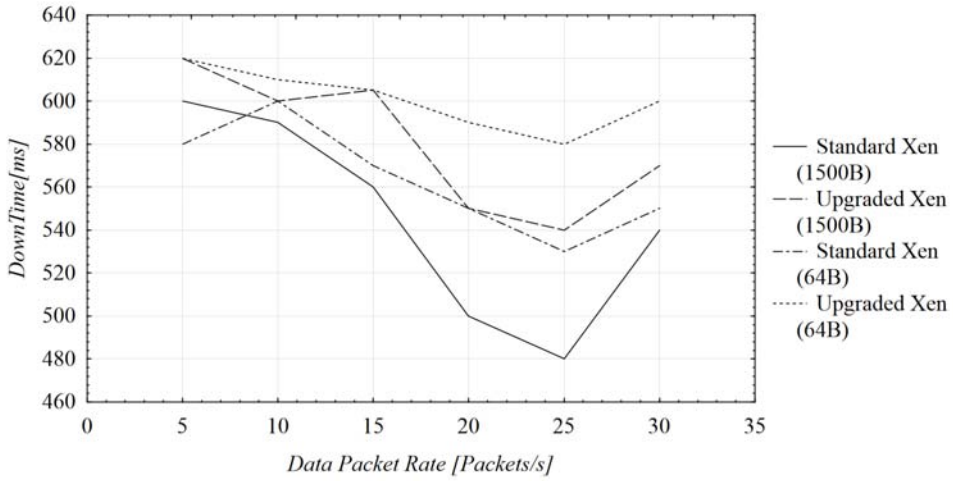

FIGURE 7. System downtime measurement for 1500 and 64 bytes MTU shows that proposed solution is only a bit slower than standard solution

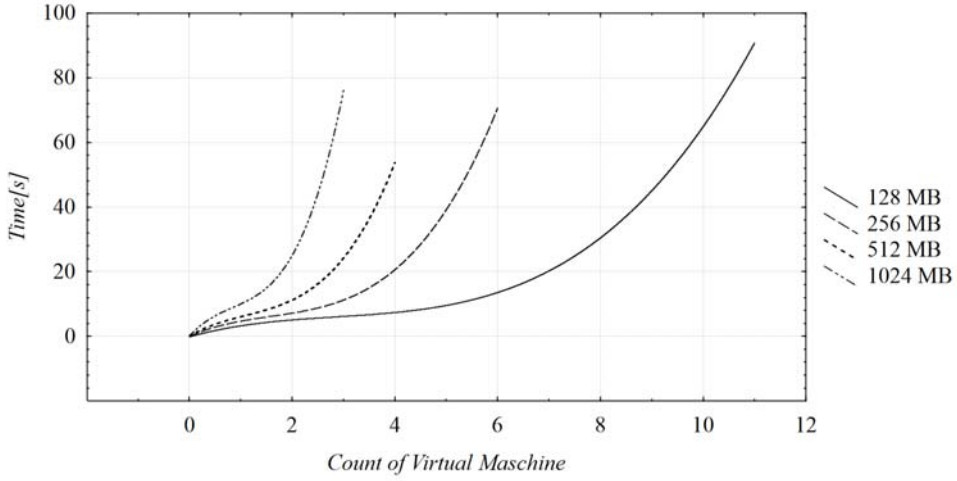

FIGURE 8. Time to migration of various counts of virtual machines depending on their operating memory sizes

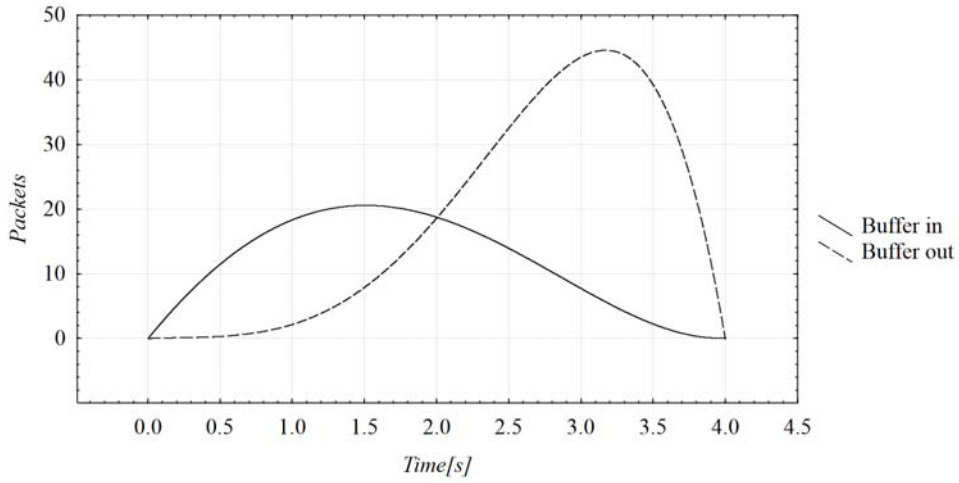

FIGURE 9. Virtual network buffers saturation during the migration phase

4.4. **Virtual network buffers saturation.** The full transparent migration process of distributed system is realized by means of the virtual network buffers. Figure 9 shows the saturation process of the in/out buffer between the source and the destination node depending on the time. This graph is made as the average of values measured in Section 4.1 and shows a gradual change.

5. **Conclusion.** We focused in our paper on the detail analysis of currently used techniques for live migration, their optimization and we also looked at basic aspects of migration of distributed systems. We proposed a scheme for their live migration, worked out prototype implementation and made some evaluations. The new migration scheme seems to be useful, i.e., at large data centers, which are running many virtual machines interconnected into virtual distributed systems.

REFERENCES

[1] D. Ruest and N. Ruest, *Virtualization*, McGraw Hill Press, 2010.
[2] C. Clark, K. Fraser, S. Hand et al., Live migration of virtual machines, *Proc. of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp.273-286, 2005.
[3] J. Jeong, S.-H. Kim, H. Kim, J. Lee and E. Seo, Analysis of virtual machine live-migration as a method for power-capping, *Journal of Supercomputing*, vol.66, no.3, pp.1629-1655, 2013.
[4] J. Oberheide, E. Cooke and F. Jahanian, *Empirical Exploitation of Live Virtual Machine Migration*, 2010.
[5] H. Jin, L. Deng, S. Wu, X. Shi and X. Pan, Live virtual machine migration with adaptive memory compression, *CLUSTER'09*, pp.1-10, 2009.
[6] *LZO Real-Time Data Compression Library*, http://www.oberhumer.com/opensource/lzo/.
[7] P. R. Wilson, S. F. Kaplan and Y. Smaragdakis, The case for compressed caching in virtual memory systems, *Proc. of the USENIX Annual Technical Conference (USENIX'99)*, pp.101-116, 1999.
[8] Gustafsson, *Optimizing Total Migration Time in Virtual Machine Live Migration*, Master Thesis, University of Uppsala, 2013.
[9] *VMware Sphere 5*, www.vmware.com/en/products/vsphere/features-vmotion.
[10] Microsoft, Hyper-V, *Configure Live Migration and Migrating Virtual Machines without Failover Clustering*, http://technet.microsoft.com/en-us/library/jj134199.aspx.
[11] G. Colouris, J. Dollimore, T. Kindberg and G. Blair, *Distributed Systems Concept and Design*, 5th Edition, Addision-Wesley, 2012.
[12] R. S. Morrison, *Architectures, Operating Systems, Parallel Processing & Programming Languages*, 2004.

[13] A. D. Kshemkalyani and M. Singhal, *Distributed Computing: Principles, Algorithms and Systems*, Cambridge University Press, 2011.
[14] Microsoft, *System Center 2012 R2*, http://www.microsoft.com/en-us/server-cloud/products/system-center-2012-r2/.
[15] VMware, *Vcenter*, http://www.vmware.com/products/vcenter-server/.
[16] OpenNebula Community, *OpenNebula*, http://opennebula.org/.
[17] OpenStack Foundation, *OpenStack*, http://www.openstack.org/.
[18] OpenQRM Community/openQRM enterprise, *OpenQrm*, http://www.openqrm-enterprise.com.
[19] J. Peng, *Comparison of Several Cloud Computing Platforms, Information Science and Engineering*, 2009.
[20] Citrix, *Citrix Xen Server*, http://www.citrix.com/products/xenserver/.
[21] Network Working Group, *Resource Reservation Protocol Specification v1*, http://tools.ietf.org/html/rfc2205.html.
[22] C. Jo, E. Gustafsson, J. Son and B. Egger, Efficient live migration of virtual machines using shared storage, *ACM Sigplan Notices*, vol.48, no.7, pp.41-50, 2013.
[23] U. Deshpande, B. Schlinker, E. Adler and K. Gopalan, Migration of virtual machines using cluster-wide deduplication, *Proc. of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2013)*, pp.394-401, 2013.
[24] P. Riteau, C. Morin and T. Priol, Shrinker: Efficient live migration of virtual clusters over wide area networks, concurrency and computation-practice & experience, *Concurrency and Computation: Practice and Experience*, vol.25, no.4, pp.541-555, 2013.
[25] H. Jin, L. Deng, S. Wu, X. Shi, H. Chen and X. Pan, MECOM: Live migration of virtual machines by adaptively compressing memory pages, *Future Generation Computer Systems*, pp.23-35, 2014.