Improving Virtual Machine Migration via Deduplication

Jake Roemer[†], Mark Groman[‡], Zhengyu Yang[§], Yufeng Wang[†], Chiu C. Tan[†], and Ningfang Mi[§]
[†] Department of Computer and Information Sciences, Temple University
[‡] Department of Computer Science and Engineering, Lehigh University
[§] Department of Electrical and Computer Engineering, Northeastern University
Email: {tuc42833,Y.F.Wang,cctan}@temple.edu, reeve.groman@lehigh.edu
yangzy1988@coe.neu.edu, ningfang@ece.neu.edu

***Abstract*—For this study the techniques of virtual machine migration are understood and the affects deduplication has on migration are evaluated. The benefits of using deduplication and compression on virtual machines show in the metric of space saved during migrating. Deduplication is computationally expensive so we evaluate how to group virtual machines with similar elements in order to improve migration. From this study, grouping virtual machines based on similar elements improves the overhead from deduplication and compression but estimates which virtual machines are best grouped together.**

***Index Terms*—Virtual Machine (VM) migration, Deduplication, Total Migration Time, Storage Space, Application Degradation, Clustering.**

I. Introduction

Cloud computing has become an ever growing part of our virtual society. Whether you are improving consumer experience or a data center strengthening their storage space and reliability, cloud computing is widely used. With so many using cloud computing as a resource, we must focus our attention on improving and optimizing this service.

Data centers use cloud computing for its generous storage system. To maintain a reliable and effective service to their costumers, data centers will often migrate between multiple Virtual Machines (VMs). This allows data centers to recover from failing VMs or to maintain an even work distribution. For virtual machine migration to ultimately benefit the user, the shift from one virtual machine to another must be seamless.

Live migration has been studied extensively to optimize various metrics in different scenarios. Virtual Machine migration for data centers are large scale, multi-gigabytes to terabytes worth of data. This makes moving VMs harder to maintain and keep efficient in both offline and online approaches. An offline approach must migrate data within a time frame of inactivity. An online approach must move data and hold network connection without the end user noticing delay. Data is also susceptible to becoming outdated or dirtied during live migration in the event of an update during transfer.

1) This study focuses on techniques used to improve the transition from one virtual machine to another in cloud computing.
2) Due to large data sets deduplication and compression has become a frequency method of reducing the amount of data needed to migrate which consequently improves migration time. This study takes an analysis of VM image snapshots and determines if deduplication will ultimately improve VM migration.
3) This study looks to find which states VM images should be in to be prime candidates for optimal migration.

II. Related Work

This study focuses on how we can improve live VM migration. We want to understand different migration techniques and convince ourselves that deduplication is worth performing on VM images. Live migration is described in [1] as well as the common techniques of pre-copy and post-copy migration [2].

An understanding of how live data can be stored and how deduplication affects virtual machine images in the cloud is studied in [3] and [4]. These papers evaluate finding similarities between VM images and explains how deduplicating these similarities will affect VM images in the cloud. We see the benefits and drawbacks of using different deduplication techniques to improve storage space and the computational cost of deduplication in [**?**] and [5].

Similar data blocks are common between virtual machines which [6] takes advantage of. The basic idea is to find and track the similar data blocks between all virtual machines. This way only a single copy of the identical data blocks need to be transferred. The identical blocks are found using deduplication and then compression is used to increase the performance of migration time. A similar approach is seen in [7] where VM images are grouped together based on deduplication. Then these groups are deduplicated with the virtual machines at the destination. Once the system is optimized it updates by looking at access patterns from the original set of virtual machines. Most migration is done under the assumption that moving virtual machines is done locally. In [8] live migration is evaluated over wide area networks (WANs) using content-based addressing. The total amount of data is reduced using deduplication to improve total migration time. Since data is being sent over a WAN, a cryptographic hash function is used to keep track of unique data and which data already exists at different locations. Once the data is distributed to the destination it goes through content-based addressing which determines the correct location for the data to be transferred.

In addition to deduplication, compression is a common way to reduce the amount of data needed to migrate, as in [9] and [10]. In [9], delta compression prevents virtual machine data from becoming outdated or dirtied during transfer. This is done by migrating a full VM first and then only migrating the changes in content to keep the migration live. In [10], the characteristics of the data content are looked at to find the correct choice of compression. Content such as many zero bytes and non-zero similarity between pages is looked for and tracked by keeping a dictionary of similar content updated. This initial evaluation is used to quicken and optimize compression of data and decompression at the destination.

III. Virtual Machine Migration

Virtualization allows multiple operating system instances to run simultaneously on one physical machine which utilizes all of the physical resources at hand. The difficulty is when moving a virtual machine from one physical machine to another without loosing connection and without any noticeable down-time. Most importantly, cloud computing services rely on this mechanic of holding an active internet connection while transferring the memory, storage and network from one physical machine to another. This means we have to worry about the bandwidth to make sure the downtime stays in a seamless time frame of milliseconds.

Storage transfers and memory data transfers hinder performance and can cause large downtime. This is due to either the size of the data being moved or the difference in live data between two virtual machines. The virtual machine needs to pause to calculate any remaining differences to keep the system safe from corrupted or degraded data and then resume on the new virtual machine which causes the milliseconds of downtime.

A. Metrics

The important metrics virtual machine migration optimizes are:

1) Total migration time is the amount of time needed to move data from source to destination. This time interval must be as small as possible to prevent the end user from noticing any delay. In the event that migration takes longer than a few milliseconds at any given time then the end user is prevented from using the virtual machine and live migration is not achieved.
2) The total storage space of the source affects how much data must be migrated. The more space that needs to be moved the longer the migration time. To achieve a better migration time the storage space can be decreased using deduplication and compression.
3) Application degradation can occur if data is updated while also being migrated. This can be prevented by shortening the migration time or adapting the delta compression technique from [9]. Even though data integrity is important it is not a common problem for most techniques.

B. Common Techniques

Two common methods of virtual machine migration are pre-copy and post-copy [2]. Pre-copy starts migrating data from source to destination while the source is still running. During this time updates can be made since the VM is still running on the source. The data which is changed is copied again and updated at destination. Once the amount of data being copied again occurs quicker than changes are made the VM is stopped. Now that the VM is stopped, no more updates can be made and the final data changes are copied over. Then the VM is resumed at the destination. The time between the virtual machine stopping and resuming is the total migration time. If small enough the user will not be able to notice that the switch was ever made.

Post-copy takes the essential data of a running virtual machine, such as CPU state and registers, and migrates it to destination first. This is enough to start the virtual machine at the destination and copy the remaining data after. As the data is copied from the source it is possible to request a part of the data that has not been transferred yet. In this event a page fault is triggered and the source will try to move the requested data. If this migration time is not small enough it is possible that data can be degraded. Pre-copy is a technique that can have a larger downtime but will keep the data up-to-date. For post-copy there is a small amount of data which is needed to start the virtual machine at the destination but data is susceptible to degradation. These techniques are used as basic migration methods and are optimized using deduplication and compression techniques in different scenarios.

To improve these techniques deduplication and compression are used to find similarities in which to cut down on and reduce the size of the data sent. These secondary operations still take time and can affect the system negatively if the overhead outweighs the saved migration time.

Sometimes there can be two virtual machines that are a better match than two other virtual machines. To prevent bad matches and optimize space saved; this study looks at the process of grouping memory images using clustering techniques.

C. Deduplication

Deduplication is the process of comparing sets of data on a binary level in order to remove identical data blocks. This technique is used to save space by eliminating redundancies. After the data has been broken down into binary we split the data into block sizes. Once split the data blocks are input for a hash function to check for a match in the hash table.

In the event of a match, a counter is incremented and the block is not stored. When no match is made the block is unique and therefore stored in the hash table. Once finished, the hash table accounts for the differences between files and markers for where the similarities exist. The redundant data can now be removed and space is saved. By minimizing the total amount of storage space the amount of time needed to migrate is also reduced.

There are a few techniques used when dealing with deduplication based on accuracy and cost of time. Whole file hashing is extremely quick since files are not broken up but compared as an entire file. This fails to be very effective in finding similar files since a small update can cause a file to be entirely unique to its previous version.

Fixed-block hashing is slower but much more accurate. For this technique, data is split up into fixed size chunks and then these chunks are compared. The benefits to this technique is that even if a file is changed only slightly we know that they are mostly similar.

D. A clustering of virtual machine images

For this study, deduplication is used to cut down on space needed to migrate virtual machines during live migration. Since companies rely on the process of live migration so the focus is on dealing with 1 physical rack; each rack containing a maximum of n virtual machine(s). We want to migrate x virtual machines from one rack to k physical racks. The data sets are virtual machines organized by operating system, running application and benchmark time found in figure 3.

We hope to achieve an understanding of how these virtual machine images relate to each other based on the space which can be saved through deduplication. We take the virtual machine images mentioned in figure 3 and run deduplication on each pair, never repeating tests and never deduplicating the same OS running the same program. The metric our deduplication tests provide is how much space can be saved between the two VM images. Using this knowledge we see patterns between the VM images and start grouping the images into a more effective process for quicker and more efficient migration.

Due to these patterns we develop rules of thumb to maximize total migration time while minimizing computation time. Once deduplication and clustering took place we noticed that the virtual machines running the same applications would save the most space between them. This fact was seen again, at a smaller amount of space savings, for virtual machines running on the same operating system and similarly for benchmark times. This creates trends which indicate that we can intuitively group virtual machines without deduplication. Through these trends we create rules of thumb that will allow us to more effectively group virtual machine images.

E. Workload Creation

Basically there are two main infrastructures in current business VM platforms: shared-storage VM networks system (SSVNS) and non-shared-storage VM networks system (NSSVNS). SSVNS has one or more centralized storage servers (online disk images, e.g, FCoE or iSCSI with VMware VMFS, shown in figure 1) accessible for each server node (offline memory images); while in NSSVNS, each node has isolated storages (offline disk and memory images). The VM migrations at these two infrastructures mainly differ in the to-be-migrated images: migration in SSVNS only needs to migrate memory images, while migration in NSSVNS has to take care of both memory images and disk images.

(1)

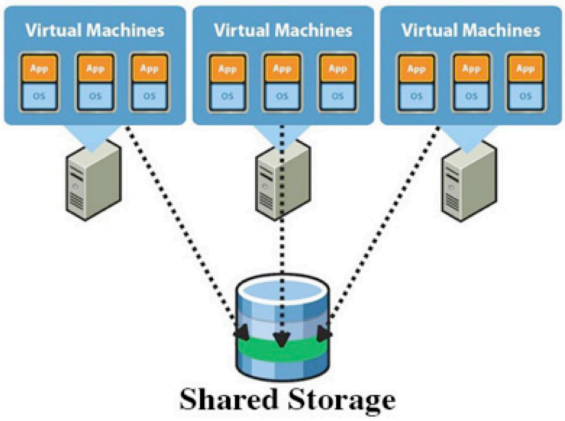

Fig. 1. A sample of shared storage networks system model.

Machine	Item	Details
Server	Model	Power Edge T310
	CPU	8 Cores CPU Intel Xeon CPU X3470 @ 2.93GHz
	HDD	250G
	Memory	8G
	OS	Linux 3.0.0-12-Server
	VM Hypervisor	Xen 4.1
VM1~4	CPU	2 Cores
	HDD	20G
	Memory	1G
	Benchmark	SPEC2006
	OS	VM1:UBUNTU12.10 / VM2:UBUNTU11.04 VM3:UBUNTU10.04 / VM4:FEDORA16

Fig. 2. Experiment Environment.

We adopt the SSVMS infrastructure in our VM platform such that only memory images are collected for deduplication analysis as well as VM migrations. We conduct experiments for collecting VM images under the open source Xen virtual machine manager (VMM) [4]. In order to emulate a variety of real runtime cases in business VM network systems, we consider a set of workloads which differ in the base operating systems or in the running applications or both of them. Table III-E shows five representative workloads used in our experiments, and Table III-E gives the specifics of the software and hardware used in our test bed.

Table 2: Control Group

Num.	OS	OS Version	Application
1	Same	Same	Different
2	Same	Different	Same
3	Same	Different	Different
4	Different	-	Different
5	Different	-	Different

In order to force frequent changes in VM images, we calculate the memory change ratios (MCRs) for each benchmark in SPEC2006. Based on our observation in table 4, we choose the top three memory change ratio benchmarks in SPEC2006[5]: BZIP2, MCF and ASTAR. They simulate three different real world workloads that modify memory heavily.

Table 3: Table of VM Images

Benchmark Name	MCR (4G Mem)	MCR (1G Mem)
401.BZIP2	22%	84%
429.MCF	21%	83%
473.ASTAR	7%	28%

BZIP2 - Compression Simulation: Compression (as well as decompression) is a commonly used operation in VMs (e.g, archive in webservers) which consumes a large amount of memory. The 401.BZIP2 is a compression benchmark based on Julian Seward's bzip2 version 1.0.3. Compression and decompression can be performed within the memory, which change 84% of the total 1G memory or 22% of the total 4G memory (shown in table 3). As a result, such a benchmark is designed to test CPU and memory only with few disk accesses.

MCF - Combinatorial Optimization Simulation: The 429.MCF is a program used for the single-depot vehicle scheduling in public mass transportation, which considers an environment with one single depot and a homogeneous vehicle fleet. This benchmark frequently changes the memory pages in order to reduce cache misses and accelerate program performance by rearranging elements of struct node and struct in the program in order, which are entirely done in the memory.

ASTAR - Artificial Intelligence Simulation: The 473.ASTAR is a computer games benchmark made by Lev Dymchenko. ASTAR includes Artificial Intelligence and Path finding. ASTAR is derived from a portable 2D path-finding library that is used in game's AI. Although compared with the previous two benchmarks, ASTAR does not consume large ratio of the memory, it is still a strongly representative application in real world.

As shown in Table 4, in the experiment, we run applications (three benchmarks) on four operating systems and get 60 memory images in total. For each OS pass (the rows in the table), we continuously run each benchmark for five times. For each run, we use the xm dump-core command to capture the memory image. The benefit of dump-core is that it does not required to turn off the VM. Instead, it just pause the VM for little seconds and save the memory file, which ensured the SLA (Service-level agreement). Moreover, in order to see both the difference of duplication degree between consistently running and non-consistently running scenarios, we on purpose reboot the physical server between different workloads (e.g, between image 5 and image 6, there is a physical reboot).

F. Deduplication Testing Framework

For our implementation we take *n* virtual machines and split them up into *k* clusters each containing a maximum of *x* virtual machines. So suppose we have 12 VMs on one physical server and we want to migrate them to 4 different servers. We create a limit of how many VMs we want per server to keep the work load evenly distributed. The value of *x* is determined by $x = \frac{n}{k}$ and the remainder is added to one of the servers with space; in our case we pick an arbitrary limit of VMs in one server.

For this study, fixed block hashing is used to perform deduplication on each set of virtual machines. We test two VMs at a time to see the amount of space saved between both. The results form a relation between each pair of virtual machines which represents the space saved between them.

We expect to see similarities between virtual machines which have one of the following elements in common: operating systems, running application and/or benchmark times. In the event these tests show no correlation between any element then no assumption can be made about the data. Therefore the overhead for deduplication or compression would have to be taken in order to improve migration. If expectations are met then we can predict that virtual machines with the same similar elements can be grouped for optimizing migration without overhead.

Once deduplicated the virtual machines are clustered for evaluation. [11] and [12] describe the simplest kind of clustering methods, called Hierarchical Methods, which construct clusters recursively by updating a cluster. This method reduces the risk of error when finding the greatest links between two virtual machines.

The three different hierarchical methods we are using for clustering are:

1) Single-linkage or nearest neighborhood. The largest edge between two nodes is chosen and then recursively the next largest until all edges are connected without circuits. After this, cut the smallest *(k-1)* edges, where *k* is the number of clusters you want, and group the remaining connections.
2) To get an even distribution of nodes we need to choose a *k* number of clusters and an *x* maximum number of nodes in a cluster. This method then repeats as nearest neighborhood but will start a new cluster if the maximum is reached for one cluster.
3) An alternate form to evenly distribute virtual machines across multiple servers is to start the *k* clusters with no node incident to more than one edge; two VMs per cluster. Once the *k* clusters are set up, the largest cluster is recursively filled until the *x* maximum number of virtual machines is reached. Then repeat the process for the remaining clusters.

IV. Evaluation

A. Deduplication of Data

The following combinations of virtual machine attributes were tested to determine common trends:

1) Similar operating systems running both the same and different applications with different benchmark times.
2) Same operating systems but different versions running both the same and different applications with different benchmark times.
3) Different operating systems running both the same and different applications with the same benchmark times.

From the data acquired from deduplication for data set 1, we noticed hierarchical method 1 is unreliable because only two VMs migrated away from the original set of VMs. For methods 2 and 3 we start to see a trend forming between the

	OS	Version	Workload 1 - BZIP2					Workload 2 - MCF					Workload 3 - ASTAR				
1	ubuntu	12.10	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	ubuntu	11.04	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
3	ubuntu	10.04	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
4	Fedora	16	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

Fig. 3. List of VM images

virtual machine similarities. We see that the virtual machines are grouped mostly based on the program they are running instead of the time the snapshots were taken. We can also notice that the virtual machines sharing the same application 1(BZIP) has the strongest relation for space saved between the operating systems. This is our first indication that applications have the largest influence on space saved from deduplication.

For data set 2, we see for hierarchical method 2 the same pattern as before where the greatest similarity between the virtual machines is the applications being run; 1(BZIP) being the greatest contributor to space saved. The remaining clusters in method 2 indicate that there is a greater correspondence between the version of the operating system than the benchmark time the snapshot was taken. This again is a clear indication that there are correlations between what the data was running and space saved.

For the last data set 3, during hierarchical method 2 we notice a trend of the most space saved between virtual machines running the same application. These tests show that there should be a rule for grouping the virtual machines together to maximize the space saved when deduplicating. The next set of tests use the data obtained for deduplication and clusters the VMs into groups.

B. Clustering of Data

Common simple methods of clustering are picking data randomly and Round Robin. We test these methods against the three hierarchical methods of clustering explained earlier. The clustering will group the virtual machines and determine the total amount of storage space after full deduplication. The hierarchical methods are expected to group virtual machines based on the similar elements tested earlier. Since both common methods have a small overhead, if either common method performs better than the hierarchical methods then groupings based on similar elements will be too computationally costly.

From figure 4 we can see picking random is the worse case since it transfers the most amount of space. Round Robin transmits as much data as hierarchical Method 2. Methods 1 and 3 are small improvements over Method 2 and Round Robin. Method 1 is not a valid method due to inefficient work distribution. You can see from this graph that the clustering methods perform better than the standard model of grouping data into clusters. From these results we can safely say grouping virtual machines is beneficial.

The following graphs perform clustering using the hierarchical methods against the intuitive rule of thumb. The

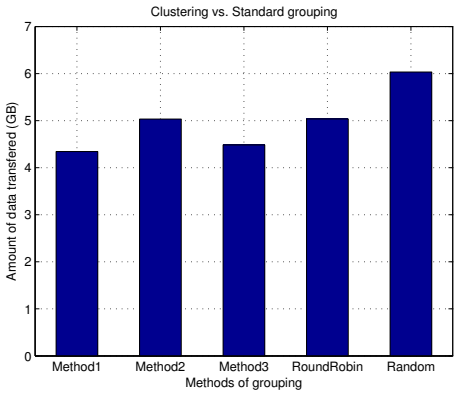

Fig. 4. Clustering Vs Standard Methods

rule of thumb created will match virtual machines based on common elements shared between them. This method has very little overhead since no extra information is needed from the virtual machines. Depending on how much space is saved will determine if virtual machines can be grouped based on similar elements alone or if a clustering method should be used in place.

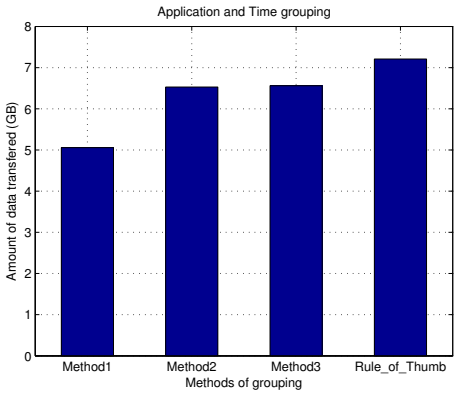

Fig. 5. Application vs. Time

Figure 5 compares varying application against varying benchmark time. The rule of thumb groups virtual machines sharing the same application first and then by operating system followed by benchmark time. The groups created from the different methods support the intuition that virtual machines sharing similar applications group together better than similar benchmark time. This graph shows that hierarchical methods perform better than rule of thumb in amount of data transferred. If you notice the rule of thumb does not outperform

the other methods but the amount of space saved is close.

From the way the clusters formed we noticed that applications were grouped together instead of benchmark time. This supports the theory that application is a better match between virtual machines than grouping by benchmark time. Hierarchical methods perform better than rule of thumb but the computation time is a lot higher.

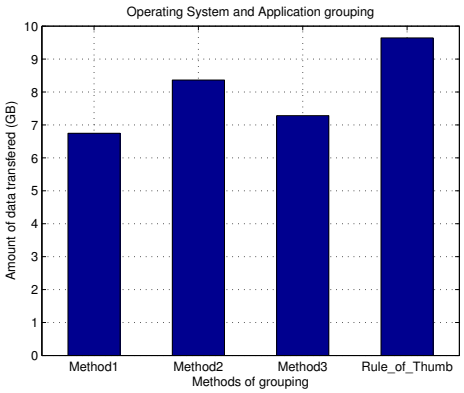

Fig. 6. Application vs. Operating System

For figure 6 we compare varying operating system against varying application. This graph shows that the rule of thumb transfers more data than the other methods. From clustering the data we find that grouping by operating system does not perform better than grouping by application. This solidifies the rule of thumb for grouping by application then operating system followed by benchmark time. The trend is that the hierarchical methods transfer less data than rule of thumb but at a higher computational cost.

V. Conclusion

This study focuses on the techniques of virtual machine migration and how to improve it. Virtual machine migration can be looked at in many different scenarios which all need the metric migration time improved. To improve migration time the amount of space needed to transfer during migration must be minimized. Optimizing storage space is achieved through deduplication and compression but both methods are computationally expensive. An alternate solution for optimizing virtual machine migration is to estimate which virtual machines should move together without the overhead of deduplication or compression.

From a collection of many studies done on virtual machine migration a trend between the data is seen. On average virtual machines which are running the same application have a higher frequency of similar blocks than virtual machines sharing the same operating system. If migration time needs to be optimized with no constraints on how long it takes to find the best matching virtual machines then deduplication and clustering should be used. Otherwise, if less than perfect matches between virtual machines can be made then grouping based on only common running application will cut overhead.

References

[1] C. C. Keir, C. Clark, K. Fraser, S. H, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *In Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI*, 2005, pp. 273–286.

[2] A. Shribman and B. Hudzia, "Pre-copy and post-copy vm live migration for memory intensive applications," in *Euro-Par 2012: Parallel Processing Workshops*, ser. Lecture Notes in Computer Science, I. Caragiannis, M. Alexander, R. Badia, M. Cannataro, A. Costan, M. Danelutto, F. Desprez, B. Krammer, J. Sahuquillo, S. Scott, and J. Weidendorfer, Eds., vol. 7640. Springer Berlin Heidelberg, 2013, pp. 539–547.

[3] C.-H. Ng, M. Ma, T.-Y. Wong, P. Lee, and J. Lui, "Live deduplication storage of virtual machine images in an open-source cloud," *Middleware 2011*, pp. 81–100, 2011.

[4] K. Jayaram, C. Peng, Z. Zhang, M. Kim, H. Chen, and H. Lei, "An empirical analysis of similarity in virtual machine images," in *Proceedings of the Middleware 2011 Industry Track Workshop*. ACM, 2011, p. 6.

[5] G. Lu, Y. Jin, and D. H. Du, "Frequency based chunking for data de-duplication," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 287–296.

[6] U. Deshpande, X. Wang, and K. Gopalan, "Live gang migration of virtual machines," in *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, ser. HPDC '11. New York, NY, USA: ACM, 2011, pp. 135–146.

[7] S. Al-Kiswany, D. Subhraveti, P. Sarkar, and M. Ripeanu, "Vmflock: Virtual machine co-migration for the cloud," in *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, ser. HPDC '11. New York, NY, USA: ACM, 2011, pp. 159–170.

[8] P. Riteau, C. Morin, and T. Priol, "Shrinker: Improving live migration of virtual clusters over wans with distributed data deduplication and content-based addressing," in *Euro-Par 2011 Parallel Processing*, ser. Lecture Notes in Computer Science, E. Jeannot, R. Namyst, and J. Roman, Eds. Springer Berlin Heidelberg, 2011, vol. 6852, pp. 431–442.

[9] P. Svärd, B. Hudzia, J. Tordsson, and E. Elmroth, "Evaluation of delta compression techniques for efficient live migration of large virtual machines," in *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ser. VEE '11. New York, NY, USA: ACM, 2011, pp. 111–120.

[10] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan, "Live virtual machine migration with adaptive, memory compression," in *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, Aug 2009, pp. 1–10.

[11] J. Grabmeier, A. Rudolph, and I. I. Gmbh, "Techniques of cluster algorithms in data mining. version 2.0," *IBM Informationssysteme GmbH*, 1998.

[12] L. Rokach and O. Maimon, "Chapter 15 clustering methods."