International Journal of Engineering Research and Modern Education (IJERME)
ISSN (Online): 2455 - 4200
(www.rdmodernresearch.com) Volume I, Issue I, 2016

AVOIDING THE DE-DUPLICATION USING CONTENT SIMILARITY AND JOB SCHEDULING ALLOCATION BASED ON WORKLOADS

N. Ramki* & A. Anitha**

* PG Scholar, Department of Master of Computer Applications, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamilnadu

** Assistant Professor, Department of Master of Computer Applications, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamilnadu

Abstract:

Experimentally, magnetic tape item has been used for database backup. With the explosion in disk capacity, it is now impossible to use disk for data backup. The Cloud storage is used for the database backup. The chunk lookup in cloud bottleneck problem that inline, chunk-based De-duplication schemes face. We perform stream De-duplication by breaking up an incoming stream into relatively large segments and De-duplicating each segment against only a few of the most similar previous segments. To identify similar segments, we use content similarity and a sparse index. We choose a small portion of the chunks in the stream as samples. Our sparse index maps these samples to the existing segments in which they occur. To reduce the task of evaluating text similarity to assessment of content similarity and use features such as bag of words to find De-duplication content. The proposed method can be allocating the resource can be based on the dependencies and the particular job execution and its weight of the each job and content similarity avoid the De-duplications.

1. Introduction:

In modern server farms, virtualization is being used to provide ever-increasing number of servers on virtual machines (VMs), reducing the number of physical machines Permission to make digital or hard copies of all or part of this work for personal or class room use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.

To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission. This approach better utilizes server resources, allowing many different operating system instances to run on a small number of servers, saving both hardware acquisition costs and operational costs such as energy, management, and cooling. Individual VM instances can be separately managed allowing them to serve a wide variety of purposes and preserving the level of control that many users want.

However, this flexibility comes at a price: the storage required to hold hundreds or thousands of multi-gigabyte VM disk images, and the inability to share identical data pages between VM instances. One approach saving disk space when running multiple instances of operating systems on multiple servers, whether physical or virtual, is to share files between them; i. e., sharing a single instance of the /usr/local file via network mount. This approach is incompatible with VM disk images; however, since the internal file structure of a VM disk image is invisible to the underlying file system.

Standard compression such as that provided by the Lempel-Ziv compression is ineffective because, while it can reduce the storage space used by a single disk image, it cannot eliminate commonalities between files. Instead, others have proposed the use of De-duplication to reduce the storage space required by the many different VM disk images that must be stored in a medium to large scale VM hosting facility. While it

seems clear that de duplication is a good approach to this problem, our research quantifies the benefits of using De-duplication to reduce the storage space needed for multiple VM disk images.

The experiments also investigate which factors impact the level of de-duplication available in different sets of VM disk images some of which are under system control (e. g., fixed versus variable-sized chunking and average chunk size) and some of which are dependent on the usage environment (e. g., operating system version and VM target use). By quantifying the effects of these factors, the results provide guidelines for both system implementers and sites that host large numbers of virtual machines, showing which factors are important to consider and the costs of making design choices at both the system and usage level.

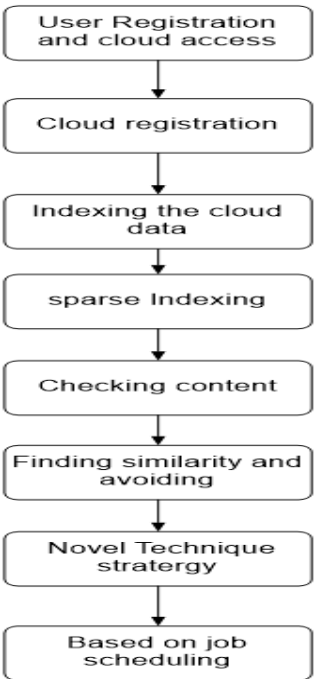


Figure 1: Flow Chart

2. Related Works:

The Effectiveness of Deduplication on Virtual Machine Disk Images:

Virtualization is becoming widely deployed in servers to efficiently provide many logically separate execution environments while reducing the need for physical servers. While this approach saves physical CPU resources, it still consumes large amounts of storage because each virtual machine (VM) instance requires its own multi-gigabyte disk image. Moreover, existing systems do not support ad hoc block sharing between disk images, instead relying on techniques such as overlays to build multiple VMs from a single “base” image. Instead, we propose the use of De-duplication to both reduce the total storage required for VM disk images and increase the ability of VMs to share disk blocks.

To test the effectiveness of De-duplication, we conducted extensive evaluations on different sets of virtual machine disk images with different chunking strategies. Our experiments found that the amount of stored data grows very slowly after the first few virtual disk images if only the locale or software con- figuration is changed, with the rate of compression suffering when different versions of an operating system or different operating systems are included. We also show that fixed length chunks work well, achieving nearly the same compression rate as variable-length chunks. Finally, we show

that simply identifying zero-filled blocks, even in ready-to use virtual machine disk images available online can provide significant savings in storage.

An Empirical Analysis of Similarity in Virtual Machine Images:

To efficiently design De-duplication, caching and other management mechanisms for virtual machine (VM) images in Infrastructure as a Service (IaaS) clouds, it is essential to understand the level and pattern of similarity among VM images in real world (IaaS) environments. This paper empirically analyzes the similarity within and between 525 VM images from a production (IaaS) cloud.

Besides presenting the overall level of content similarity, we have also discovered interesting insights on multiple factors affecting the similarity pattern, including the image creation time and the location in the image's address space. Moreover, we found that similarities between pairs of images exhibit high variance, and an image is very likely to be more similar to a small subset of images than all other images in the repository. Groups of data chunks often appear in the same image. These image and chunk "clusters" can help predict future data accesses, and therefore provide important hints to cache placement, eviction, and prefetching.

Efficiently Storing Virtual Machine Backups:

Physical level backups offer increased performance in terms of throughput and scalability as compared to logical backup models, while still maintaining logical consistency. As the trend toward virtualization grows, virtual machine backups (a form of physical backup) are even more important, while becoming easier to perform. The downside is that physical backup generally requires more storage, because of file system meta-data and unallocated blocks.

De-duplication is becoming widely accepted and many believe that it will favor logical backup, but this has not been well studied and the relative cost of physical vs. logical on de duplicating storage is not known. In this paper, we take a data-driven approach using user data to quantify the storage costs and contributing factors of physical backups over numerous generations. Based on our analysis, we show how physical backups can be as storage efficient as logical backups, while also giving good backup performance.

Allow Bandwidth Network File System:

Users rarely consider running network file systems over slow or wide-area networks, as the performance would be unacceptable and the bandwidth consumption too high. Nonetheless, efficient remote file access would often be desirable over such networks particularly when high latency makes remote login sessions unresponsive. Rather than run interactive programs such as editors remotely, users could run the programs locally and manipulate remote files through the file system. To do so, however, would require a network file system that consumes less bandwidth than most current file systems.

Avoiding the Disk Bottleneck in the Data Domain Deduplication File System:

Disk-based De-duplication storage has emerged as the new-generation storage system for enterprise data protection to replace tape libraries. De-duplication removes redundant data segments to compress data into a highly compact form and makes it economical to store backups on disk instead of tape. A crucial requirement for enterprise data protection is high throughput, typically over 100 MB/sec, which enables backups to complete quickly. A significant challenge is to identify and eliminate duplicate data segments at this rate on a low-cost system that cannot afford enough RAM to store an index of the stored segments and may be forced to access an on-disk index for every input segment. This paper describes three techniques employed in the

production Data Domain De-duplication file system to relieve the disk bottleneck. These techniques include: (1) the Summary Vector, a compact in-memory data structure for identifying new segments; (2) Stream-Informed Segment Layout, a data layout method to improve on-disk locality for sequentially accessed segments; and (3) Locality Preserved Caching, which maintains the locality of the fingerprints of duplicate segments to achieve high cache hit ratios. Together, they can remove 99% of the disk accesses for De-duplication of real world workloads. These techniques enable a modern two-socket dual-core system to run at 90% CPU utilization with only one shelf of 15 disks and achieve 100 MB/sec for single-stream throughput and 210 MB/sec for multi-stream throughput.

Building a High-Performance Deduplication System:

Modern De-duplication has become quite effective at eliminating duplicates in data, thus multiplying the effective capacity of disk-based backup systems, and enabling them as realistic tape replacements. Despite these improvements, single-node raw capacity is still mostly limited to tens or a few hundreds of terabytes, forcing users to resort to complex and costly multi-node systems, which usually only allow them to scale to single digit petabytes. As the opportunities for De-duplication efficiency optimizations become scarce, we are challenged with the task of designing De-duplication systems that will effectively address the capacity, throughput, management and energy requirements of the peta scale age.

In this paper present our high-performance De-duplication prototype, designed from the ground up to optimize overall single-node performance, by making the best possible use of a node's resources, and achieve three important goals: scale to large capacity, provide good De-duplication efficiency, and near raw disk throughput. Instead of trying to improve duplicate detection algorithms, we focus on system design aspects and introduce novel mechanisms that combine with careful implementations of known system engineering techniques.

In particular, we improve single node scalability by introducing progressive sampled indexing and grouped mark and sweep, and also optimize throughput by utilizing an event-driven, multi-threaded client/server interaction model. Our prototype implementation is able to scale to billions of stored objects, with high throughput, and very little or no degradation of De-duplication efficiency.

Ceph: A Scalable, High-Performance Distributed File System:

Ceph, a distributed file system that provides excellent performance, reliability, and scalability. Ceph maximizes the separation between data and metadata management by replacing allocation tables with a pseudo random data distribution function (CRUSH) designed for heterogeneous and dynamic clusters of unreliable object storage devices (OSDs). We leverage device intelligence by distributing data replication, failure detection and recovery to semi-autonomous OSDs running a specialized local object file system. A dynamic distributed metadata cluster provides extremely efficient metadata management and seamlessly adapts to a wide range of general purpose and scientific computing file system workloads. Performance measurements under a variety of workloads show that Ceph has excellent I/O performance and scalable metadata management, supporting more than 250,000 metadata operations per second.

PVFS: A Parallel File System for Linux Clusters:

As Linux clusters have matured as platforms for low cost, high-performance parallel computing, software packages to provide many key services have emerged, especially in areas such as message passing and networking. One area devoid of support, however, has been parallel file systems, which are critical for high performance

I/O on such clusters. We have developed a parallel file system for Linux clusters, called the Parallel Virtual File System (PVFS). PVFS is intended both as a high-performance parallel file system that anyone can download and use and as a tool for pursuing further research in parallel I/O and parallel file systems for Linux clusters. PVFS and present performance results on the Chiba City cluster at Argonne.

The provide performance results for a workload of concurrent reads and writes for various numbers of compute nodes, I/O nodes, and I/O request sizes. We also present performance results for MPI-IO on PVFS, both for a concurrent read/write workload and for the BTIO benchmark. We compare the I/O performance when using a Myrinet network versus a fast ethernet network for I/O-related communication in PVFS. We obtained read and write bandwidths as high as 700 Mbytes/sec with Myrinet and 225 Mbytes/sec with fast ethernet.

3. Proposed Work:

The De-duplication method best suited to protect data in cloud. This process De-duplicates data both across backups and within backups and does not require any knowledge of the backup data format. The job can be system allocation can be performed for the batch jobs with the sequence of job allocation. And the content similarity is used for the de-duplications process and filtering the De-duplication content. In the time interval, the job can be finished with the effective resources then allocation can be in the order sequences. The included automates filtering, to help an analyst in cloud with similar content by designating of Data duplication can be easily removed by the content similarity algorithm. The workloads can be categorized as per the order of the job work load can be assigned. The scheduling can be maintained as per the sequence of the job within the time interval the particular job can be executed.

A. User Registration and Cloud Access: Access users only to have authentication process before registration, Authentication process is always occurred prior to mobility management process included location registrations and service delivery, and it also ensures network resources are accessed by authorized clients and prevents resources from any illegal client or damage. Before the registration of cloud services to ensure whether the client is an authenticated or not to access cloud server. We can ensure the information stored in the cloud is used judiciously by the responsible stakeholders as per the service level agreements.

B. Indexing the Cloud Data: The based on requirements to prepare the dataset in avoid de-duplication content. Indexing is nothing but consists of structured and unstructured format. Unstructured format is an unarranged format. Sparse Indexing is based on the reference format and capturing the repeated words queries. Indexing converts the unarranged format into structured arranged format. This may be avoid the problem of delay during searching. Sparse Indexing are used to quickly locate data without having to search every database based on the queries is accessed.

C. Finding & Avoiding Similarity: Content similarity detection is typically performed by means of De-duplication, which is broadly classified into static and content defined. Static approaches split the input data in to equally sized chunks, which are then compared among each other. In order to identify and eliminate duplicates. While simple and fast, static approaches suffer from misalignment issues (i.e insertions or deletions lead to the impossibility to detect duplicates).Comparison phase quantifies the degree of similarity between indexing pairs belonging to the same data. And blocking the De-duplication chunks using novel techniques. Novel technique strategy aimed at reducing the user labeling effort in large scale De-duplication tasks.

D. Allocating the Workloads Job Management Based on Content Similarity:

The resource can be allocated based on the dependencies of the each job. Based on the dependencies the resource can be allocated. The Content Similarity is a statistical methods to categorize a De-duplication and blocked the adjustable levels of granularity. We cultrate the data set, so that it contains only one representation of each sequence for quantifying and comparative studies. The included automates filtering, to help an analyst in cloud with similar content by designating of Data duplication can be easily removed by the content similarity algorithm.

4. Experimental Analysis and Results:

Implementation is often used in the tech world to describe the interactions of elements in programming languages. In Java, where the word is frequently used, to implement is to recognize and use an element of code or a programming resource that is written into the program.

One aspect of implementing an interface that can cause confusion is the requirement that to implement an interface, a class must implement all of the methods of that interface. This can lead to error messages due to insufficient implementation of methods. In general, the syntactical requirements of implementation and other tasks can be a burden for developers, and mastering this is part of becoming an in-depth user.

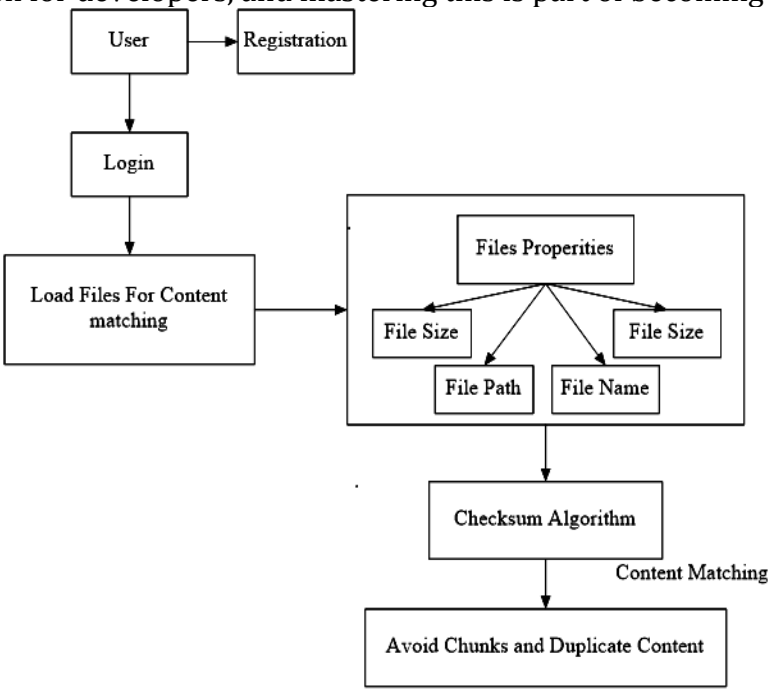


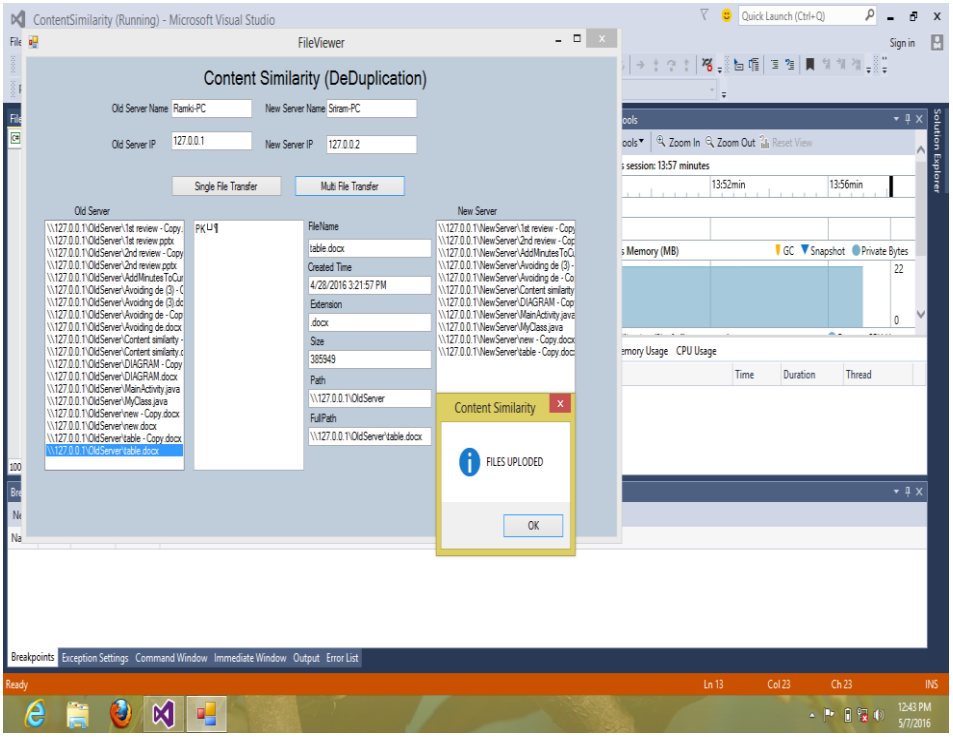
Figure 2: System Architecture

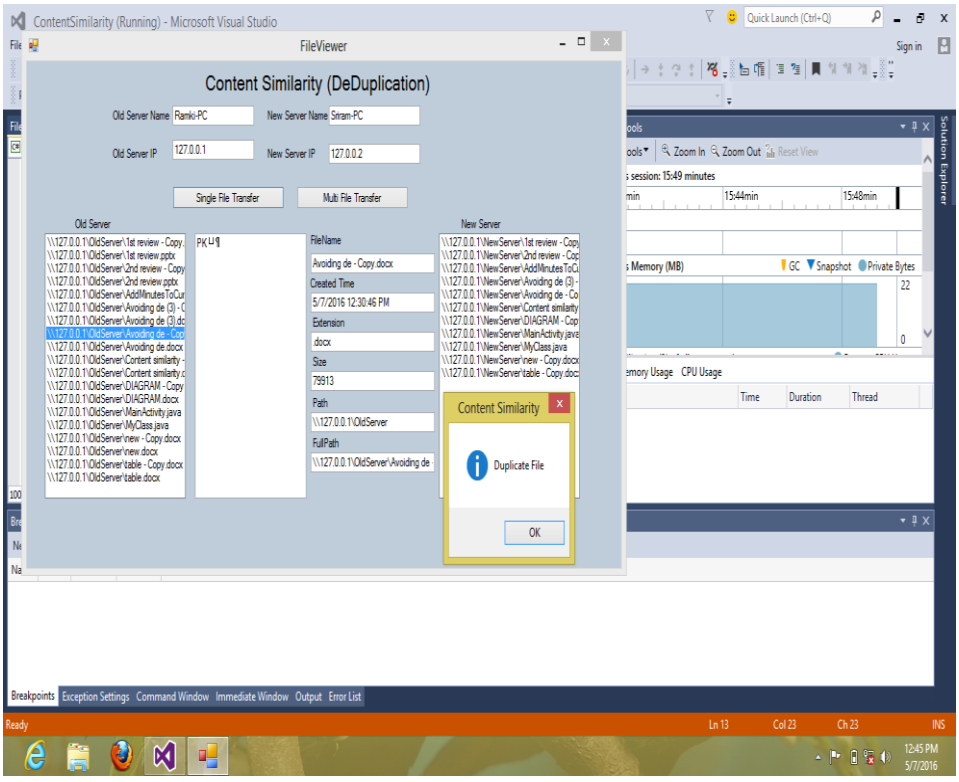
There are multiple storage services for a user to store data. Meanwhile, to avoid the problem produced by the centralized “trusted” third party, the responsibility of SeDas is to protect the user key and provide the function of self-destructing data. The brief structure of the user application program realizing storage process. In this structure, the user application node contains two system clients: any third-party data storage system (TPDSS) and SeDas. The user application program interacts with the SeDas server through SeDas’ client, getting data storage service.

Implementation is the process of translating design specification in to source code. The primary goal of implementation is to write source code and internal implementation. So that conformance of code to its specification can be easily verified,

So that debugging, testing and modification are eased. The source is developed with clarity, simplicity and elegance.

The coding is done in a modular fashion giving such importance even to the minute detail so, when hardware and storage procedures are changed or now data is added, rewriting of application programs is not necessary. To adapt or perfect use must determine new requirements, redesign generate code and test exiting software/hardware. Traditionally such task when they are applied to an existing program has been called maintenance.





5. Conclusion and Future Enhancement:

We explored the impact of many factors on the effectiveness of De-duplication. We showed that package installation and language localization have little impact on De-duplication ratio. However, factors such as the base operating system. The Linux distribution can have a major impact on De-duplication effectiveness. Thus, we recommend that hosting centers suggest "preferred" operating system distributions for their users to ensure maximal space savings. If this preference is followed subsequent user activity will have little impact on De-duplication effectiveness. We found that, in general, 40% is approximately the highest De-duplication ratio if no obviously similar VMs are involved.

In future work we plan to explore several promising avenues. First, we did not explore what happens when the groups are not operating simultaneously and/or access common content at different times. How to leverage and anticipate such De-synchronizations can provide further potential for improvement. Second, our approach treats all chunks individually, both in terms of advertisements and exchanges. Thus, it would be interesting to understand and exploit correlations between chunks.

6. References:

1. K. Jin and E. L. Miller, "The effectiveness of De-duplication on virtual machine (VM) disk images," in Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, ser. SYSTOR '09. Haifa, Israel: ACM, 2009, pp. 7:1–7:12.
2. K. R. Jayaram, C. Peng, Z. Zhang, M. Kim, H. Chen, and H. Lei, "An empirical analysis of similarity in virtual machine images," in Middle ware '11: Proceedings of the Middleware 2011 Industry Track Workshop. Lisbon, Portugal: ACM, 2011, pp. 6:1–6:6.
3. R. Schwarzkopf, M. Schmidt, M. R¨udiger, and B. Freisleben, "Efficient storage of virtual machine images," in Science Cloud '12: Proceedings of the 3rd Workshop on Scientific Cloud Computing Date. Delft, the Netherlands: ACM, 2012, pp. 51–60.
4. Muthitacharoen, B. Chen, and D. Mazi`eres, "A low-bandwidth network file system," SIGOPS Oper. Syst. Rev., vol. 35, no. 5, pp. 174–187, Oct. 2001.
5. M. Rabin, "Fingerprinting by random polynomials," Center for Research in Computing Technology, Harvard University, Tech. Rep. TR-CSE-03-01, 1981.
6. B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain De-duplication file system," in FAST'08: Proceedings of the6th USENIX Conference on File and Storage Technologies. San Jose, USA: USENIX Association, 2008, pp. 18:1–18:14.
7. C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W.Kilian, P. Strzelczak,J. Szczepkowski, C. Ungureanu, and M. Welnicki,"Hydrastor:a scalable secondary storage," in FAST '09: Proceedings of the 7thconference on File and storage technologies. San Francisco, USA: USENIX Association, 2009, pp. 197–210.
8. F.Guo and P. Efstathopoulos," Building a high-performance De-duplication system," in USENIXATC'11: Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference. Portland, USA: USENIX Association, 2011, pp. 25–39.
9. B. Nicolae, "Towards Scalable Checkpoint Restart: A Collective Inline Memory Contents Deduplication Proposal," in IPDPS '13: The 27thIEEE International Parallel and Distributed Processing Symposium, Boston, USA, 2013, pp. 19–28.
10. Z. Shen, Z. Zhang, A. Kochut, A. Karve, H. Chen, M. Kim, H. Lei, andN. Fuller, "Vmar: Optimizing i/o performance and resource utilizationin the cloud," in Middleware

’13: Proceedings of the 14th ACM/I-FIP/USENIX International Middleware Conference, vol. 8275. Springer Berlin Heidelberg, 2013, pp. 183–203.

11. K. Razavi, A. Ion, and T. Kielmann, “Squirrel: Scatter hoarding vm image contents on iaas compute nodes,” in HPDC ’14: The 23rd ACM International Symposium on High-performance Parallel and Distributed Computing. Vancouver, Canada: ACM, 2014, pp. 265–278.
12. R. Koller and R. Raju, “I/o deduplication: Utilizing content similarity to improve i/o performance,” in FAST ’10: Proceedings of the USENIX File and Storage Technologies. USENIX Association, 2010, pp. 211–224.
13. B. Mao, H. Jiang, S. Wu, Y. Fu, and L. Tian, “Read-performance optimization for deduplication based storage systems in the cloud,” Trans. Storage, vol. 10, no. 2, pp. 6:1–6:22, Mar. 2014.
14. U. Deshpande, X. Wang, and K. Gopalan, “Live gang migration of virtual machines,” in HPDC ’11: Proceedings of the 20th International Symposium on High Performance Distributed Computing. San Jose, USA: ACM, 2011, pp. 135–146.
15. S. Al-Kiswany, D. Subhraveti, P. Sarkar, and M. Ripeanu, “Vmflock: Virtual machine co-migration for the cloud,” in Proceedings of the 20thInternational Symposium on High Performance Distributed Computing, ser. HPDC ’11. San Jose, USA: ACM, 2011, pp. 159–170.