Speeding Up VM Image Distribution for Cloud Data Centers

Choonhwa LEE[1], Hyungseock LEE[1], Eunsam KIM[2]
[1]*Division of Computer Science and Engineering, Hanyang University, Seoul 133-791, Rep. of Korea*
[2]*Department of Computer Engineering, Hongik University, Seoul 121-791, Rep. of Korea*
eskim@hongik.ac.kr

***Abstract*—An efficient scheme of virtual machine image dissemination, which takes up a major portion of VM provisioning time, is key to ensuring that the service can be provisioned and delivered to clients in a timely manner. This paper presents a novel peer-to-peer protocol to efficiently distribute virtual machine images in a cloud data center. The primary idea of our proposal is to expedite the peer-to-peer content delivery by taking advantage of a high level of similarity that oftentimes exists among different VM images. The efficacy of our protocol is validated through an evaluation that demonstrates substantial performance gains over existing approaches.**

***Index Terms*—computer networks, content distribution networks, distributed computing, peer-to-peer computing, protocols.**

I. Introduction

While some early cloud offerings turned out to be very successful, there remain several technical challenges for a wider acceptance. One main obstacle to clear is a long provisioning latency; whatever resources it intends to offer, a cloud service provider should be able to deliver them in a timely fashion, as if there exists an unlimited pool of the resource instances that is ready to be tapped at any time. It is known that it oftentimes takes several minutes to provision a VM in the cloud data center [1]. On receipt of a VM service request, a virtual machine instance is launched based on OS images downloaded from an image server of the data center network. The download takes the lion's share of the provisioning delay, because the size of the images often ranges up to several or even tens of gigabytes. In addition, the data center network has to deal with a massive number of the instance requests, as a cloud datacenter is being challenged to meet the ever-growing demand of high performance computing [2-3]. Therefore, an efficient dissemination scheme of VM images is key to ensuring satisfactory user experiences with the IaaS offerings.

The sheer volume of VM provisioning traffic, which can easily overwhelm a datacenter network, calls for an innovative approach. The problem could be tackled by designing a new data center network architecture, so that the network can provide ample bandwidth for inter-host communication [4]. In a complementary effort, the utmost use might be made of the networking infrastructure by employing new content delivery schemes. Representative of such efforts are peer-to-peer content dissemination protocols and deduplication schemes that have proven effective in delivering contents over the Internet [5-7]. In this paper, we investigate the potential of integrating the P2P delivery and deduplication schemes to speed up virtual machine provisioning in the cloud data center network that is often stretched thin to deal with a high volume of traffic. Our approach is to enable faster image distribution by allowing cross-image sharing in a peer-to-peer fashion.

This research was supported by the MSIP(Ministry of Science, ICT&Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2016- H8501-16-1006) supervised by the IITP(Institute for Information&communications Technology Promotion) and by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2016R1D1A1A09917396).

II. VM Image Distribution Schemes

The problem of VM provisioning latency might be mitigated by repeating what was done for traditional paging systems to enhance OS performance. The execution time of VMs could be reduced by prioritizing image blocks that are going to soon be needed, when making a scheduling decision of network transfers. Also, a VM might be started as soon as a minimal set of image blocks are acquired with additional blocks being downloaded on the fly as needed. However, these conventional techniques alone would not be enough to deal with a massive number of VM instance requests (e.g., at the scale of hundreds of thousands of VMs) in a short period of time. Cloud data center networks would be quickly overwhelmed by the deluge of the traffic, as the image files are often up to tens of GBs long. The research community has been exploring viable solutions to the problem in the past several years. Prominent technologies to the rescue include image deduplication [8-9], P2P content sharing [10-11], and network topology-awareness [8], [12].

Data deduplication techniques have proven effective in reducing the amount of information that needs to be stored or transferred [7], [13-14]. More specifically, files are chopped into data blocks, called chunks, using either fixed-sized or variable-sized chunking methods, so that redundant ones can be skipped for storage or network transfer. A good portion of the deduplication research was directed to efficient ways to detect the redundancy; a number of deduplication techniques can be categorized into white-box [15] and black-box schemes [7], [13]. The former blindly handles data as a bit sequence that is fed to cryptographic hash functions like MD-5 or SHA-1 for hash values. It contrasts the latter where some structural or semantic information of the data is made use of for deduplication.

According to recent similarity studies, VM images share a good amount of common blocks with others [13], [16-17]. For instance, content similarity between close versions of

Digital Object Identifier 10.4316/AECE.2016.04002

the same OS can be as high as 60%, as seen in the case of CentOS 6.1 and 6.2 [13]. Therefore, it should not be a surprise to witness numerous research efforts to investigate deduplication techniques for the cloud environments, including VM delivery, scheduling, launch, and migration [8], [13-14], [18].

Another trend of VM provisioning research is found at the idea of leveraging P2P swarming protocols for VM image distribution that has actually been around for some time [10], [19]. Since those early efforts where their primary focus was on the adoption itself of P2P BitTorrent-like protocols for VM image dissemination in the cloud data centers, the idea has developed into more advanced protocols like P2P-based cross-image chunk distribution schemes [20-21]. But they did not fully explore the possibility of the idea in that the potential of the cross-image sharing was studied to some limited extent.

III. P2P-Based Cross-Image Distribution Protocol

This paper investigates the possibility of deduplication combined with peer-to-peer content delivery. Our idea is that if the boundaries between P2P swarms can be removed, peers would be able to take advantage of common blocks shared across the swarms of different image files and versions. Then, the heightened availability of image blocks should speed up the image distribution process. Our approach is motivated by the recent studies of VM images for representative cloud offerings that reported a significant degree of similarity at image and block level [13], [17]. In order to enable cross-image swarming, we can no longer rely on image block offsets as a block id in the way we did for peer-to-peer content sharing. Instead, image blocks are named and identified by hash-based fingerprints [8], [21]. By comparing the hash values, identical blocks can now be detected regardless of what image files they belong to, which effectively removes swarm boundaries and consequently enables cross-image deduplication.

Our attention was also brought to a pattern by which a group of image chunks (i.e., a chunk cluster) often appear repeatedly in different images. It is more so, when the images are related to each other, e.g., close versions of the same OS. VM provisioning latency could be more adequately dealt with, if we can exploit the presence of chunk clusters over different images. We employ Merkle tree as a means to quickly identify and communicate the presence of such chunk clusters. A Merkle tree, also known as a hash tree, is a tree structure where an intermediate node is labelled with the hash of the concatenated labels of its children nodes, and a leaf node is labeled with a hash value of its corresponding data block [22]. The data structure has been in use because of its ability of efficient and secure verification of partial content being downloaded. The tree is utilized by our cross-image distribution scheme to communicate chunk clusters of VM images among peers rather than a means for integrity checking.

A. Hash Tree-Based Cluster Detection

Our idea of Merkle tree-based similarity detection, whether a single chunk or a group of chunks, is illustrated in Fig. 1. The figure shows that three peers join a cross-image swarm to download their respective images that are all different. Hash trees corresponding to each image are depicted in Fig. 1 (b). Suppose that peer 1 is in the process of downloading an image made up of three clusters: *a*, *b*, and *c*. Peer 2's image consists of clusters *d*, *b*, and *e*. Compositions of the images that others possess can be efficiently learned through hash tree exchanges with the neighbors. For instance, peer 2 learns that its second section *b* is identical to that of peer 1 by exchanging a single node with label *b* that represents the chunk cluster. In other words, peer 2 infers that the cluster is shared between the two images from the fact that there exists an identical hash value in its Merkle tree. In a similar way, the fact that peer 3 owns chunk cluster *e* gets known to peer 2. Peer 2 can now download chunk cluster *b* from peer 1 and cluster e from peer 3, while downloading cluster *d* from other sources. In short, Merkle tree allows a group of continuous image chunks to be identified with a single hash exchange. Such an exchange can be used for advertising the presence of a chunk cluster as well as for soliciting a cluster from neighbor peers.

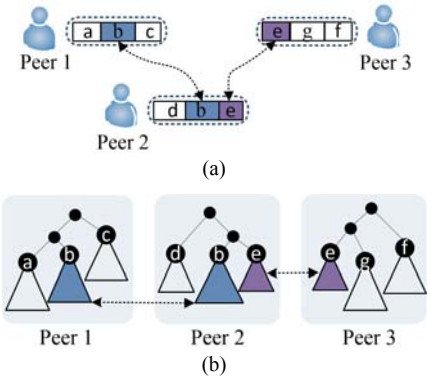

Figure 1. Merkle tree-based similarity detection. (a) Three peers in cross-image swarming. (b) Corresponding Merkle trees.

Fig. 2 illustrates the architectural design of our peer-to-peer cross-image chunk distribution approach. The scheme is designed to exploit cross-image content similarity to ease the distress of massive image distribution traffic on the cloud data center network. First, VM images are divided into chunks using an appropriate chunking method. Chunks are then fingerprinted by hash functions like MD5 or SHA-1. (1) Having followed the usual bootstrapping procedure of BitTorrent-like swarming protocols, which results in the image's chunk information being transferred as part of the swarming metadata, (2) peer 1 contacts a tracker also serving as an image server of the data center in this particular configuration. The tracker replies with a list of peers, participating in a swarm of the same image and/or similar images, along with an image similarity matrix. The similarity matrix provides some hints on which image swarms for the joining peer to start with. Then, (3) by performing our cross-image distribution protocol, peer 1 engages itself in chunk trading with other peers. Chunk fingerprints, i.e., nodes of the Merkle tree, are exchanged among swarming partners. As a result, peer 1 figures out which parts of the image are in common with peer 2. A desired chunk cluster can be indicated and acquired from the neighbor by sending a corresponding node in the hash tree. Summing up, our approach enables cross-image swarming to allow common cluster transfers among different images,

while minimizing the overhead to identify common clusters, resultantly expediting VM image distribution process.

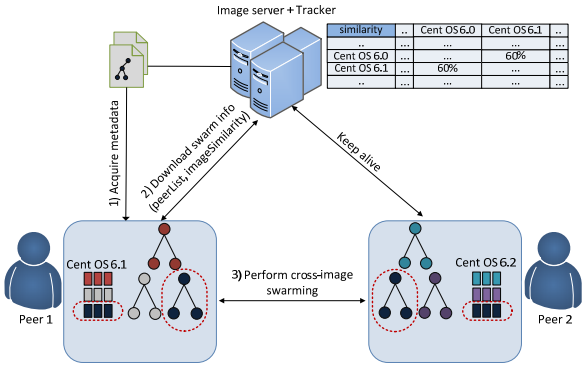

Figure 2. Cross-image distribution of VM images.

B. XID Protocol

Our swarming protocol is named as XID (cross-image distribution) protocol. Based on the idea of peer-to-peer content sharing across VM image files, XID protocol first advertises a hash value that represents either an individual chunk or a group of chunks. A neighbor peer may respond back with a hash value to indicate what is asked for. Alternatively, the neighbor could decide to explore other parts of the hash tree in search of nodes of interest. As diagrammed in Fig. 3, the protocol employs four different message types: Hash, Probe, Request, and Block.

(1) Peer 1 starts off with the root *a* of its hash tree. It packs the top part of the tree in a Hash message to send to one of its neighbors, peer 2. Subsequently, peer 1 advertises the next part of the tree, rooted at the node labelled with *b*, in another Hash message. The advertisement continues to cover other parts of the tree, scanning the tree in a top-down fashion.

(2) When peer 2 receives a Hash message, it searches its own hash tree for a possible match with hash values in the message. When peer 2 sees a match at node *c*, it realizes that there exists an overlapping part between the two trees of peer 1 and peer 2.

(3) From this point on, peer 2 takes an active role to seek further information by sending a Probe message for a particular part of the tree, i.e., the subtree rooted at node *c*. The probe message is answered by a corresponding Hash message from the other side that contains part of the tree starting at node *c*.

(4) Suppose that peer 2 finds that it misses image chunks beneath node *d*. Then, the peer requests all the data blocks in the subtree by sending a single hash value *d* in a Request message. Peer 1 replies back with a series of Block messages that convey the chunk cluster rooted at *d*.

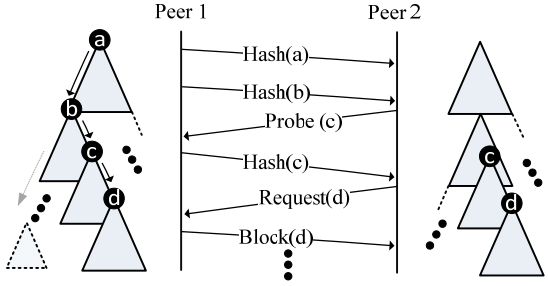

Figure 3. Operation of XID protocol.

In short, our XID protocol employs Merkle tree to identity a cluster of chunks across different image files. Peers are able to advertise and request a group of chunks using a single hash value. As a result, chunks can be traded across swarms with minimal overhead. Fig. 4 roughly sketches XID protocol in terms of messages exchanged among peers.

```
protocol X-IMAGE CHUNK DISTRIBUTION
  h : height of subtree to advertise in a HASH message
  HT : hash tree representation of an image under consideration

  // Advertisements, not shown in the code below, are performed
  // by a separate background thread using HASH messages. The ads
  // scan the hash tree, HT, in a top-down fashion.

  while message ← receiveMessage() do
    switch (message.getType())
      case HASH:
        if further exploration is needed
          nextHashSet ← message.getWhatToExplore(HT)
          foreach nhash in nextHashSet
            sendMessage(PROBE, nhash)
        else if chunk fetch is needed
          dataHashSet ← message.getWhatToFetch(HT)
          foreach dhash in dataHashSet
            sendMessage(REQUEST, dhash)
      case PROBE:
        phashSet ← message.getProbedHash()
        stree ← HT.packProbedSubTree(phashSet, h)
        sendMesssage(HASH, stree)
      case REQUEST:
        rhash ← message.getRequestedHash()
        dBlocks ← HT.packDataBlock(rhash)
        sendMessage(BLOCK, dBlocks)
      case BLOCK:
        dBlocks ← message.getDataBlock()
        HT.updateImage(dBlocks)
    end switch
  end while
```

Figure 4. Cross-image chunk distribution algorithm.

C. An Improvement of XID Protocol

As discussed, Merkle trees are used as a means to efficiently detect common chunk clusters. However, XID protocol may suffer from a long leading delay before actual chunk trading. Oftentimes, it may not be until a long series of Hash message exchanges that the protocol discovers a point of interest in the hash tree of the other side, because the advertisement starts off from the top of the tree.

This leading delay of the protocol can be avoided by employing Bloom filters. A set of image chunks a peer owns can be communicated to its neighbors using a Bloom filter in a single step. More specifically, the whole hash tree can be encoded into a Bloom filter that is sent to neighbors at the beginning. On receipt of the Bloom filter, a peer checks whether the advertising neighbor possesses some image chunks of interest. If it is the case, then the peer replies back with a probe or chunk request message. The protocol follows the same procedure as the base version from this point on. This improved version is named XID-BF hereafter in the paper.

IV. Performance Evaluation

In order to prove the efficacy of our cross-image distribution protocol, we performed a simulation study that compares its performance with those of other alternatives. Our simulator is built on PeerSim P2P simulator (http://peersim.sourceforge.net).

A. Simulation Setup and Alternative Protocols

The data center network for our simulation is structured as a three-layer hierarchy of 32×8×8 nodes; 32 nodes housed in a rack are interconnected by a top-of-rack switch that is in turn aggregated by a second-tier switch. Eight aggregation switches are connected to one of eight tier-1 switches at the top of the hierarchy. Effective bandwidth of each tier is set to 100MB, 500MB, and 2GB, respectively, reflecting the oversubscribed network architecture. Simulation runs involve 1,024 peers spread randomly over 2,048 physical hosts.

The distribution rate and size of OS images considered for our simulation are summarized in Table I. The distribution of particular versions was determined based on the popularity of Web server OSes [13]. Similarity among OS images was reported in several recent studies, one of which is used for our simulation study [23]. Finally, images are divided into a sequence of chunks of 4KB for the simulation, and SHA-1 hashes of 20 bytes are used as a chunk id.

TABLE I. DISTRIBUTION AND SIZE OF OS IMAGES

OS Image	Distribution (%)	Size (GB)
CentOS 5.0	21	1.3
CentOS 6.3	14	1.6
Fedora 15	8	1.8
Fedora 16	9	1.9
Fedora 17	9	2
Ubuntu 11.04	1	1
Ubuntu 11.10	2	1
Ubuntu 12.04	3	1
Windows	33	4

XID protocol is compared with alternative image distribution schemes for a data center network such as BT, VDN, HL, and XID-BF. As discussed earlier in the paper, XID-BF indicates the Bloom filter-based improvement over the base XID protocol to reduce signaling traffic among peers.

- *BT case* represents a BitTorrent-like delivery protocol where an individual swarm is formed for each and every image. Since swarming is performed within the same image, a sequence number of image chunks is enough to identify a particular chunk. Therefore, the protocol incurs less signaling traffic for chunk identification. For example, given an image of 1GB and the chunk size of 4KB, the chunk bitmap of the protocol is 256Kbits long. The chunk signaling overhead of 20byte long SHA-1 hashes can grow up to 5MB for VDN, HL, XID, and XID-BF protocols.
- *VDN protocol* promotes data access locality and minimizes redundant block transfers for VM image distribution [8]. A central piece of the protocol is an indexing server that tracks image chunks present at its descendant nodes. A node queries its index server for meta-information on which node to contact for a specific chunk. Then, the server redirects the enquirer preferably to one of its descendants that has the requested chunk. A set of indexing servers are organized in a hierarchical fashion so as to reflect the underlying data center network structure. As a result, chunk transfer traffic and time can be minimized by localizing a chunk request within the smallest possible range of the hierarchy.
- *HL protocol* lets a peer exchange chunk hashes, instead of the P2P case's chunk offsets, with its neighbors to communicate what blocks it has. It can be viewed as similar to a peer-to-peer chunk exchange scheme by which common blocks from different VM images are shared [21]. A peer can participate in multiple swarms at the same time to accelerate the process of VM image acquisition, enabling parallel chunk downloads from different images. A downside of this protocol is that it incurs larger overhead compared to XID protocol, since a hash value has to be associated one-to-one with a particular chunk.

B. Performance Results

Initially, the data center network is populated to have OS images according to the distribution in Table I. For our simulation runs, 100 requests for each VM image are issued to nodes chosen randomly. As guided by the tracker, a peer engages itself in swarming with other peers preferably with virtual machine images closer to the image in question. An average time for 100 downloads of each VM image is compared in Fig. 5. As expected, BT case suffers the most, because it confines itself to an isolated swarming of the target image. Cross-image sharing leads to good performance for the rest of the protocols with XID and XID-BF protocols showing the best results. In the case of CentOS 5.8, the most popular, in our simulation setup, the average times to complete a VM image download are 15.7, 19.3, 24, 39.9, and 76 seconds for XID-BF, XID, HL, VDN, and BT, respectively.

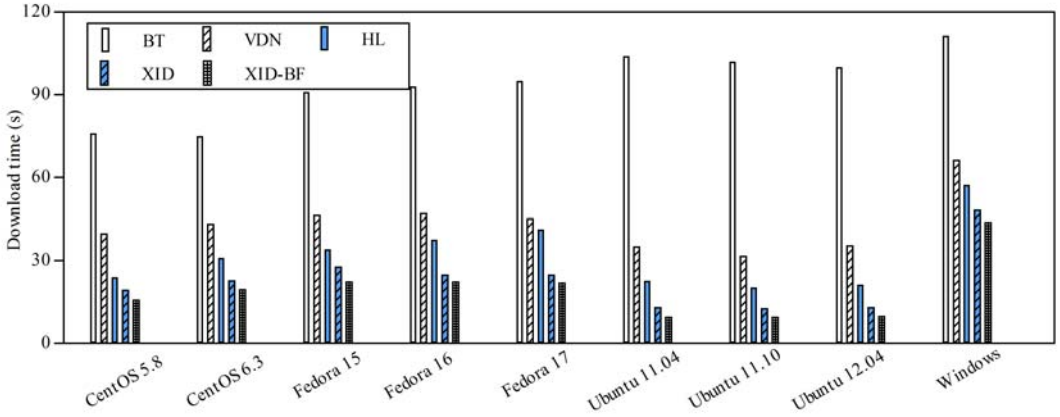

Figure 5. Download times per image.

To understand the performance results presented in Fig. 5, we look at the sources of image chunks. Fig. 6 plots the breakdown of image chunk sources for CentOS 5.8. According to where they originate from, image chunks can be classified into three categories: intra, inter, and server. Intra-chunks indicate chunks downloaded from the same image file on other peers, while inter-chunks are the ones from other images than CentOS 5.8 image. Server category indicates the portion of image chunks that originate from a dedicated image server of the data center network. The percentage of intra, inter, and server cases for XID protocol is 46:34:20, while the numbers are 44:37:19 for XID-BF protocol. VDN protocol has the breakdown of 53%, 27%, and 20% for the intra, inter, and server category, respectively. HL protocol shows a composition of 47:31:22 from the three sources. Without cross-image swarming supported, 26% of BT protocol's chunks are pulled from the image server, and the rest comes from other peers. Better content availability for cross-image schemes is credited with making a difference in the performance as shown in Fig. 5. By being able to fetch more chunks from nearby peers rather than the image server or distant peers, cross-image swarming protocols ease the strain typically placed on the upper tiers of the hierarchical data center network.

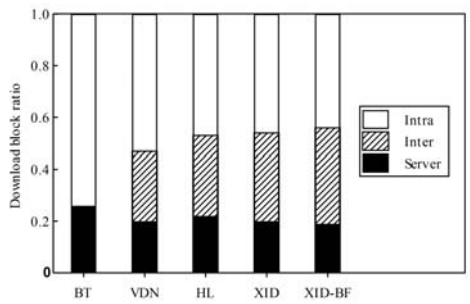

Figure 6. Breakdown of image chunk sources.

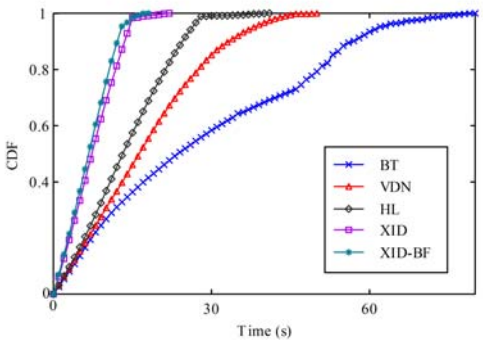

Figure 7. Image download progress.

It is evident from the results thus far that the contribution from cross-image chunks ends up becoming a big boost for overall performance compared with a single image distribution of BT protocol. We further compare the protocols in terms of their download progress rate, as time goes by. As shown in Fig. 7, XID-BF is able to complete its download of CentOS 5.8 image faster than any others in about 18 seconds, which is closely followed by 22 seconds of XID protocol. It took 42, 50, and 80 seconds for HL, VDN, and BT, respectively.

Performance differences among cross-image protocols can be explained by the way peers identify image chunks and the way to communicate such information. Chunks are individually identified by using their fingerprints in the case of VDN and HL protocols. In contrast, a cluster of chunks can be collectively indicated by a single hash for XID protocols. XID-BF achieves further optimization by compressing the fingerprints into a Bloom filter.

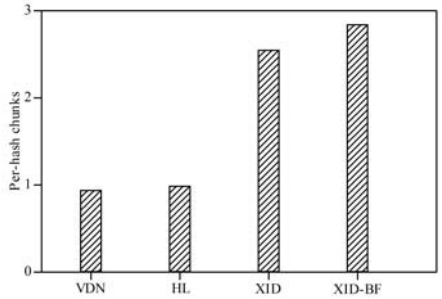

Figure 8. Number of image chunks per hash.

Fig. 8 measures the overhead of the protocols in terms of their chunk naming efficiency. The graph shows the number of chunks per hash value for each protocol. VDN is able to download 0.96 chunks for one hash, while HL has an edge over it. XID and XID-BF do much better to obtain 2.2 and 2.6 chunks at the expense of a single hash. This result reveals that performance gains of XID and XID-BF protocols come from the ability to identify a set of image chunks by one fingerprint hash. As a result, the protocols can operate with lesser overhead, which further accelerates the download process by allowing otherwise wasted network resources to be used for actual chunk downloads.

V. Related Work

Many research efforts pursued higher efficiency in VM image delivery by avoiding or minimizing redundant data access and transfer. One notable approach is to perform deduplication with the help of block translation maps generated beforehand when VM images were created [18]. It was also shown that the amount of VM provisioning traffic can be reduced, if a VM instance is scheduled on a physical host with an image close to the one in question [13]. Additionally, deduplication of common image chunks was proven beneficial for VM migration support [14], [23].

Detection of identical chunks across different images can be made possible by employing hash-based fingerprints [7-8], [18], [21]. Beyond the fingerprint-based deduplication where hash values are used as a chunk id across different files and versions, researchers went an extra mile to make further improvements. For instance, chunk fingerprint signaling overhead can be reduced by avoiding full hash value exchanges, when not absolutely necessary [24]. The protocol achieves savings in terms of the bit length of individual hashes being transferred, while our XID scheme saves hash overhead at the granularity of a group of hashes. Another interesting work is found where a file system structure is leveraged to enhance deduplication performance. More specifically, simply a directory name can be mentioned to indicate all the files and sub-directories in that directory, when a directory is copied between two machines

[25]. The idea might be viewed as similar, in spirit, to ours in that it communicates a set of files using a single directory name, instead of individually naming all the files in the directory. But it requires some knowledge of the file system structure, unlike XID protocol that need not understand any semantics of VM images.

On the other hand, there has also been research on P2P-based distribution schemes of VM images. Since early efforts of adopting BitTorrent protocols for VM image distribution [10], [19], the idea develops into further sophisticated schemes like cross-image distribution scheme [11], [20-21]. While some focus on efficient bandwidth allocations of an image server across different swarms [11], [20], the swarm boundary can be removed by employing fingerprint-based chunk ids [21]. Hash keys as inter-swarming ids allow cross-image chunk trading, which effectively consolidates all the swarms into a single swarm where chunks from different image files can be advertised and exchanged. This protocol corresponds to HL protocol in the paper. Our XID protocol goes further by advertising and requesting a cluster of image chunks rather than individual chunks, which enables even more efficient cross-image chunk sharing with much lesser overhead.

VI. Conclusion

This paper proposes a VM image distribution protocol for a cloud data center which leverages both deduplication and peer-to-peer content dissemination to speed up virtual machine provisioning. Our approach enables cross-image swarming in an efficient way that requires lesser signaling overhead among peers to detect and identify chunk clusters common to different image files. A performance study demonstrates that our proposal outperforms the state-of-the-art protocols by 34% in terms of download completion time. As for chunk naming efficiency, it shows an improvement of a factor of 1.7 over the leading-edge approach.

References

[1] M. Mao and M. Humphrey, "A performance study on the VM startup time in the cloud," in Proc. 5th IEEE International Conference on Cloud Computing (CLOUD), Honolulu, HI, USA, Jun. 2012, pp.423-430. doi:10.1109/cloud.2012.103

[2] W. Xiong and L. Chen, "HiGIS: An open framework for high performance geographic information system," Advances in Electrical and Computer Engineering, vol.15, no.3, pp.123-132, August 2015, doi:10.4316/AECE.2015.03018

[3] N. Stojanovic and D. Stojanovic, "High performance processing and analysis of geospatial data using CUDA on GPU," Advances in Electrical and Computer Engineering, vol.14, no.4, pp.109-114, November 2014, doi:10.4316/AECE.2014.04017

[4] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in Proc. ACM SIGCOMM 2008 Conference on Data Communication, Seattle, WA, USA, Aug. 2008, pp.63-74. doi:10.1145/1402958.1402967

[5] C. Lee, S. Kim, and E. Kim, "Expediting P2P video delivery through a hybrid push-pull protocol," Advances in Electrical and Computer Engineering, vol.15, no.4, pp.3-8, November 2015, doi:10.4316/AECE.2015.04001

[6] C. Ashwin, R. Bharambe, and V. Padmanabhan, "Analyzing and improving a BitTorrent network's performance mechanisms," in Proc. 25th IEEE International Conference on Computer Communications (INFOCOM), Barcelona, Spain, Apr. 2006, pp.1-12.

[7] A. Muthitacharoen, B. Chen, and D. Mazieres, "A low-bandwidth network file system," in Proc. 18th ACM Symposium on Operating Systems Principles (SOSP), Banff, Canada, Oct. 2001, pp.174-187. doi:10.1145/502034.502052

[8] C. Peng, M. Kim, Z. Zhang, and H. Lei, "VDN: virtual machine image distribution network for cloud data centers," in Proc. 31st IEEE International Conference on Computer Communications (INFOCOM), Orlando, FL, USA, Mar. 2012, pp.181-189. doi:10.1109/infcom.2012.6195556

[9] C. Lee, S. Kim, and E. Kim, "A deduplication-enabled P2P protocol for VM image distribution," IEICE Transactions on Information and Systems, vol.E98-D, no.5, pp.1108-1111, May 2015. doi:10.1587/transinf.2014EDL8180

[10] J. Reich, O. Laadan, E. Brosh, A. Sherman, V. Misra, J. Nieh, and D. Rubenstein, "VMTorrent: scalable P2P virtual machine streaming," in Proc. 8th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT), Nice, France, Dec. 2012, pp.189-300. doi:10.1145/2413176.2413210

[11] X. Leon, R. Chaabouni, M. Sanchez-Artigas, and P. Garcia-Lopez, "Smart cloud seeding for BitTorrent in datacenters," IEEE Internet Computing, vol.18, no.4, pp.47-54, Jul.-Aug. 2014. doi:10.1109/MIC.2014.43

[12] C. Raiciu, M. Ionescu, and D. Niculescu, "Opening up black box networks with CloudTalk," in Proc. 4th USENIX Conference on Hot Topics in Cloud Computing, Boston, MA, USA, Jun. 2012.

[13] S. Bazarbayev, M. Hiltunen, K. Joshi, W. H. Sanders and R. Schlichting, "Content-based scheduling of virtual machines (VMs) in the cloud," in Proc. 33th IEEE International Conference on Distributed Computing Systems (ICDCS), Philadelphia, PA, USA, Jul. 2013, pp.93-101. doi:10.1109/icdcs.2013.15

[14] U. Deshpande, B. Schlinker, E. Adler, and K. Gopalan, "Gang migration of virtual machines using cluster-wide deduplication," in Proc. 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Delft, Netherlands, May 2013, pp.394-401. doi:10.1109/ccgrid.2013.39

[15] D. Reimer, A. Thomas, G. Ammons, T. Mummert, B. Alpern, and V. Bala, "Opening black boxes: using semantic information to combat virtual machine image sprawl," in Proc. 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, Seattle, WA, USA, Mar. 2008, pp.111-120. doi:10.1145/1346256.1346272

[16] K. Kin and E. Miller, "The effectiveness of deduplication on virtual machine disk images," in Proc. 2009 Israeli Experimental Systems Conference, Haifa, Israel, May 2009.

[17] K. R. Jayaram, C. Peng, Z. Zhang, M. Kim, H. Chen, and H. Lei, "An empirical analysis of similarity in virtual machine images," in Proc. 12th ACM/IFIP/USENIX International Middleware Conference, Lisbon, Portugal, Dec. 2011. doi:10.1145/2090181.2090187

[18] Z. Shen, Z. Zhang, A. Kochut, A. Karve, H. Chen, M. Kimand, and H. Lei, "VMAR: optimizing I/O performance and resource utilization in the cloud," in Proc. 14th ACM/IFIP/USENIX International Middleware Conference, Beijing, China, Dec. 2013. doi:10.1007/978-3-642-45065-5_10

[19] Z. Chen, Y. Zhao, X. Miao, Y. Chen, and Q. Wang, "Rapid provisioning of cloud infrastructure leveraging peer-to-peer networks," in Proc. 29th IEEE International Conference on Distributed Computing Systems (ICDCS) Workshops, Montreal, Canada, Jun. 2009, pp.324-329. doi:10.1109/icdcsw.2009.35

[20] D. Wu, Y. Zeng, J. He, Y. Liang, and Y. Wen, "On P2P mechanisms for VM image distribution in cloud data centers: modeling, analysis, and improvement," in Proc. 4th IEEE International Conference on Cloud Computing Technology and Science, Taipei, Taiwan, Dec. 2012, pp.50-57. doi:10.1109/CloudCom.2012.6427568

[21] B. Nicolae, A. Kochut, and A. Karve, "Discovering and leveraging content similarity to optimize collective on-demand data access to IaaS cloud storage," in Proc. 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Shenzhen, China, May 2015, pp.211-220. doi:10.1109/ccgrid.2015.156

[22] R. C. Merkle, "A digital signature based on a conventional encryption function," Lecture Notes in Computer Science, vol.293, pp.369-378, 1988. doi:10.1007/3-540-48184-2_32

[23] H. Lai, Y. Wu, and Y. Cheng, "Exploiting neighborhood similarity for virtual machine migration over wide-area network," in Proc. 7th IEEE International Conference on Software Security and Reliability, Washington, D.C., USA, Jun. 2013, pp.149-158.

[24] J. Barreto, L. Veiga, and P. Ferreira, "Hash challenges: stretching the limits of compare-by-hash in distributed data deduplication," Information Processing Letters, vol.112, no.10, pp.380-385, May 2012. doi:10.1016/j.ipl.2012.01.012

[25] N. Jain, M. Dahlin, and R. Tewari, "TAPER: tiered approach for eliminating redundancy in replica synchronization," in Proc. 4th USENIX Conference on File and Storage Technologies (FAST), San Francisco, CA, USA, Dec. 2005, pp.281-294.