BIDS: Bridgehead-Employed Image Distribution System for Cloud Data Centers

Zhongzhao Wang[1], Yuebin Bai[1,⋆], Kun Cheng[1], Jihong Ma[2], Duo Lv[3], Yuanfeng Peng[4], and Yao Ma[1]

[1] School of Computer Science, Beihang University, Beijing 100191, China
[2] Handan Polytechnic College, Handan 056000, China
[3] Department of Computer Science, Arizona State University, Tempe, AZ 85281, USA
[4] Department of Computer Science, Pennsylvania University, Philadelphia, PA 19104, USA

Abstract. To provide elastic cloud services with QoS guarantee, it is essential for data centers to provision Virtual Machine(VM) instances rapidly. Due to bandwidth bottleneck of centralized model, the P2P-like distribution schemes are recently adopted. However, most of them just focus on the transmission speed, but ignore the impact on network bandwidth, especially for cloud data centers which are connected by Wide Area Network(WAN). In this paper, we propose a bridgehead-employed VM image distribution system(BIDS), which aims to minimize the repetitive data flows of WAN while speeding up the image distribution. In BIDS, we also design a version based image sharing mechanism, which tries to make a balance between efficiency and management complexity. Besides, we implement a Remote Management Console(RMC). The final evaluation shows that BIDS is of high efficient and low overhead.

Keywords: virtual machine(VM), image distribution, bridgehead, cloud data centers.

1 Introduction

Generally speaking, cloud data centers keep plenty of VM images. For good system elasticity and scalability, it is critical for the cloud data centers to provision VM images fast, even in the case of massive concurrent requests. In order to eliminate the bottleneck of conventional centralized model, researchers recently proposed BitTorrent-like distribution model[1,2]. This model relieves the burden on storage servers, but it just allows VM instances that start from the same image files to share chunks. Prior studies[3,4] have shown that different VM images could still have lots of common chunks. Then Peng et al.[5] proposed a cross-image distribution model, called VDN. VDN is a complete sharing model. It means that image chunks can be shared among any VM instances regardless

⋆ Corresponding author.

C.-H. Hsu et al. (Eds.): NPC 2014, LNCS 8707, pp. 269–280, 2014.

of image types. But we must be aware of the fact that VDN had made the system indexing unit, eg., *lookup*, *publish*, etc, changed from images to chunks. So VDN has a complex index structure and expensive computing cost. In addition, all current known distribution models mainly focus on distribution speed, but ignore the impact on network bandwidth. We all know that cloud data centers usually consist of multi data centers which are deployed in different regions and connected by WAN. Because WAN often refers to low bandwidth and long delay, it's inexpedient to treat WAN and Local Area Network(LAN) equally in the image distribution system.

In this paper, we propose a bridgehead-employed VM image distribution system(BIDS), which aims to minimize the repetitive data flows of WAN while speeding up the image file distribution. Different from VDN and other similar paradigms, BIDS has several novel features. First, we introduce bridgehead mechanism in the distribution model. Here we assume that the cloud data centers consist of multi Independent Data Centers(IDCs) and all these IDCs are connected via WAN. In each IDC, we specify some Designated Bridgehead(DB) hosts to decrease the bandwidth consumption of inter-IDCs. Second, we propose version-based image sharing scheme, which tries to make a balance between efficiency and management complexity. And *Delta files* are used to speed up distribution and optimize storage of each physical host. Third, we implement a Remote Management Console to facilitate the management of image resources in cloud data centers.

The rest of the paper is organized as follows. Sect. 2 describes the design principle of BIDS and Sect. 3 presents the detailed implementation. Sect.4 evaluates our proposed model. We discuss the related work in Sect. 5, followed by conclusions in Sect. 6.

2 Design Principle of BIDS

In designing the BIDS distribution framework, our primary objective is to maximize the amount of VM image data that is available within one IDC or on nearby IDCs when the image is needed.

2.1 Execution Mechanism of Bridgehead Mode

We all know that WAN has a characteristic of low bandwidth and long delay. So when one IDC own a image file, it's improper to allow the hosts in other IDCs to "download" the image data arbitrarily. We should find a way which can speed up image file distribution while minimizing WAN bandwidth consumption. In this paper, we introduce the concept of bridgehead. In each IDC, we specify some particular hosts, called DBs. Here the DB is just a normal host except for the ability of sharing image data with other IDCs. In BIDS, the ordinary hosts are forbidden to exchange image data with other IDCs' hosts directly. When a ordinary host must have to obtain image data from other IDCs, it will send request to DB(s) firstly. Here the DBs form the upper-level distribution

network. All the inter-IDC image data transmission is accomplished via this network. Thus, the inter-IDC image data exchange can be completely restricted within DBs. The waste of WAN bandwidth caused by plenty of repetitive data flows can be avoided.

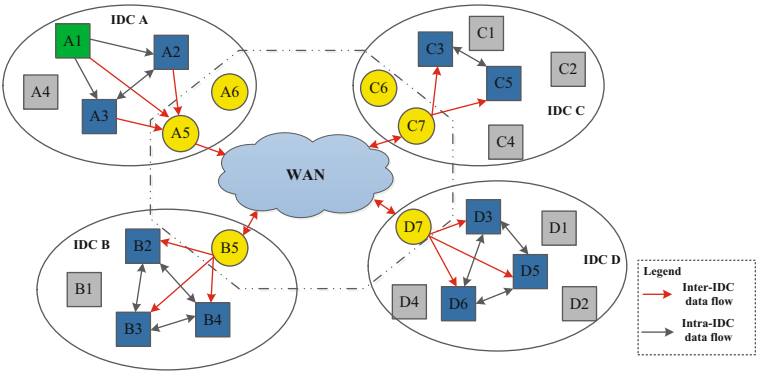


Fig. 1. The brief diagram of bridgehead_employed distribution mode

Now, we briefly illustrate the bridgehead mechanism by an example. Figure 1 shows a skeleton diagram of BIDS distribution process. The host A5, A6, B5, C6, C7, D7 are DB nodes and they form a upper-level network. Now suppose that host A1 own a image file X, host A2, A3, B2, B3, B4, C3, C5, D3, D5, D6 need to get image X. For A2, A3, they belong to the same IDC with A1, so they can get image data from A1 directly. Meanwhile, A2, A3 can share the image data they have already obtained. In such scenario, host A5, A6 are just normal hosts. They make no sense for image distribution. For B2, B3, B4, they can't get image data from A1 directly. In this case, the DB node B5 has to communicate with A5 for image data. Then host B2, B3, B4 get image data from B5. Of course, they can also share the obtained data with each other. Note that the data transmission between DBs, between DB and hosts can proceed in parallel. Analogously, IDC C, IDC D have similar processing. Here, host C7, D7 may be able to obtain image file X from multi DBs simultaneously.

As you can see from Fig. 1, A6 and C6 are DB hosts, but they don't participate in the distribution of image X. They can be viewed as normal hosts in this scenario, but they perhaps play a role of DB in other image distribution. In BIDS, the distribution of one image usually corresponds to a special hierarchical network view. In this view, part of DBs form upper-level network. A DB host may play a role of DB in one view, but act as a normal host in another view. The role of DB depends on the host_list constructed by GIS(General Index Server, detail in Sect. 2.2). This dynamic, non-immobilized scheme can improve distribution efficiency, and at the same time, avoid performance bottleneck problem caused by static scheme.

2.2 Efficient Collaborative Sharing

In cloud data centers, metadata management is one key factor of affecting the performance and efficiency. Here, the metadata consists of the list of hosts who have certain images. In BIDS, in order to manage massive metadata, we set up a particular server, called GIS. In GIS, it stores information about hosts, images, and the relationships between hosts and images, between IDCs and hosts. Besides these, GIS also needs to keep the information about *Delta files*, more details can be found in Sect.2.3. Figure 2 illustrates the work flow of *Publish* and *Lookup*.

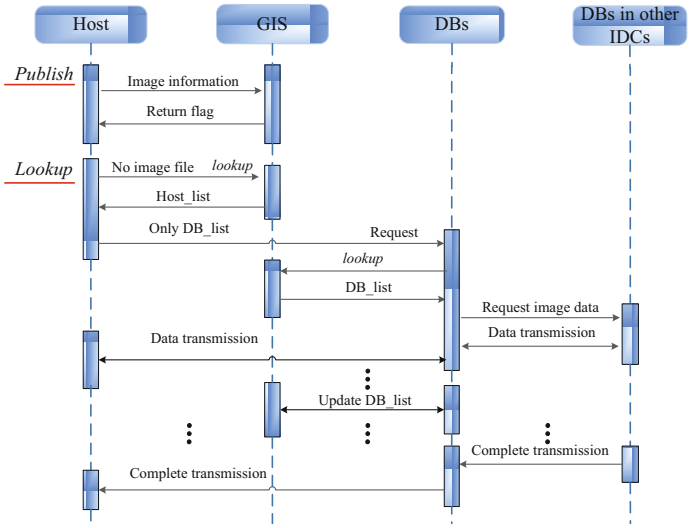


Fig. 2. The sequence char of *Publish* and *Lookup*

In BIDS, when a host, including DB, has obtained a new image, or upgraded an old image, it will execute the *publish* process. Then GIS will do some update operation. For another case, when a host, including DB, needs to get a image, it will execute the *lookup* process. In Fig. 2, we can see that the *lookup* result just contains DB hosts. This informs the requester that the request image is stored in other IDC(s). So the requester send request(s) to the specified DB(s). After that, the data transmission between DBs, between DB(s) and host begin. During the transmission, the DB(s) may communicate with GIS periodically to get the latest host information. Note that centralized structure may turns into the bottleneck of reliability. But it can be resolved easily by mature duplex backup technology.

Distribution Security. For the hosts within one IDC, because they are deployed in LAN, the malicious nodes outside the network are unable to interact

with these hosts. In addition, the host themselves can also reject the local requests because of authorization failed, invalid request, or just in order to guarantee the user service. For inter-IDC distribution, because all the DB nodes are designated by administrators with unique DB IDs, it's impossible for malicious nodes to pose as DBs to grab image files.

2.3 Image Version Management

Crossing-image sharing already become an indispensable part of improving performance. But allowing data sharing among any images, like VDN[5] does, often brings up high system overhead. In another aspect, Satyanarayanan et al.[6] show that if a set of images is based on the same major version of operating system, the similarity is high(more than 50%). Peng.c et al.[5] have concluded that the most popular images can contribute to more than 85% of all instances. So it's quiet common that users' VM requests may mainly focus on several types of images. Here each type of image may contain many versions.

Consequently, we propose a version-based image sharing scheme. In this scheme, VM instances that belong to different versions can share their data. Here, we use *Delta file* to record the difference between images that derived from the same base images, but belong to different versions. In this way, a VM instance can quickly upgrade to newer version with just one *Delta file*. And the host can quickly build a new image with just one *Delta file* if it already owns some version of the base image. Of course, it is also feasible for a host to run multi VM instances with just one image file and a series of delta files. In BIDS, *Delta file* information, relationships between *Delta files* and image versions are stored in GIS. Thus, when a host needs to upgrade its image to certain version, it can quickly find which *Delta file* is needed and where to find the *Delta file*.

Image Storage. In BIDS, we employ a distributed storage mechanism. We set up a local storage server in each IDC. These servers act as image "providers". Note that the DB and storage server play a similar role to some extent. The DB can totally act as a storage server and we have done in this way for system testing. However, if the DB is not employed for storage, its useless, outdated, or duplicate image files can be removed by administrators via RMC.

2.4 Remote Management Console

In order to facilitate the management of all the image files, we design a management console. Note that RMC is mainly used for image file management, not a sophisticated resource management platform, like RHEV[7]. To be specific, RMC mainly achieve the following functions.

- Network Information Statistics: In order to facilitate the administrator to know running status of cloud data centers, RMC communicates with GIS periodically to maintain some global statistic information. Alternatively, RMC display these information by the graph way.

– Distribution Control: RMC could control hosts remotely, like limiting "upload", "download" speed, switching sharing status. Besides, RMC could also be used to designate or change IDCs' DBs, designate some hosts to *publish* images, or manipulate hosts to "download" certain image files automatically.
– Image Management: In BIDS, we adopt flexible storage mechanism. Images may be stored in ordinary hosts and DBs. After long-time running, the hosts may store some old, or useless images. RMC can be used to remove images according to certain policies, like oldest, rerely used, or just randomly.

3 Components Implementation of BIDS

In this section, we present the details of components implementation of BIDS.

3.1 Architecture of GIS

GIS is one of key component of BIDS. It holds the running information of the whole system and makes corresponding responses for each arrived request. Figure 3 shows the main function modules of GIS.

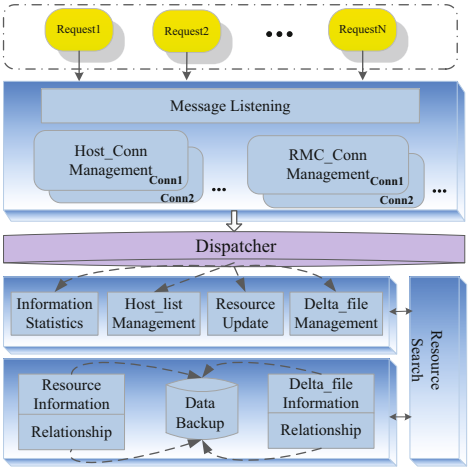


Fig. 3. Architecture of GIS

Host Conn and RMC Conn Modules are mainly used for handling interactions with other components. These two modules need to maintain the information of each received connection. Normally, these two modules may deal with plenty of connection requests concurrently. Resource Search Module is responsible for locating resource position rapidly. Here, we propose an improved hash function which bases on the implementation of Python Dictionary. In Host_list Management Module, we design lots of filter policies which apply to pick out the optimal

host list. With the host list, the requester can obtain image data with a minimum cost. Information Statistics Module mainly in response to the RMC's remote commands. The other modules are responsible for managing images, hosts, *Delta files* and image versions information.

3.2 Architecture of Host Service Part

This component is the core of BIDS, all the real image data distribution is accomplished via this part. At runtime, this component needs to communicate with GIS, RMC and other hosts. Figure 4 shows the major modules of this part.

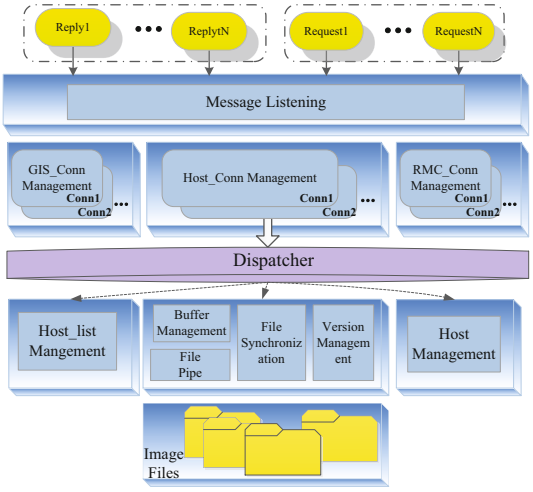


Fig. 4. Architecture of Host Service Part

Similar to SIN, we implement three Conn Modules. Besides, we implement Node Management Module in response to operation instructions sent by RMC. Host_list Management Module can identify the locally stored image information, and *publish* the image information to the GIS if necessary. It also can manage the host list received from GIS and guide the host where to obtain the required image data. File Synchronization Module mainly achieve:(i)build delta file; (ii) merge for the target file. For Version Management Module, it is mainly used to control procedure of version operation. Buffer Management and File Pipe Modules accomplish memory mapping mechanism: the file data is mapped into the memory pages directly. This method can make user process access file data directly and avoid inefficient data copy between user space and kernel space like the conventional file read mode does.

3.3 Architecture of RMC

In BIDS, RMC is implemented based on WEB using MVC design pattern. Its framework can be broadly divided into five layers: View Layer, Model Layer, Service Layer and Persistence Layer. Figure 5 shows the hierarchy architecture of RMC. Using such hierarchy, on the one hand, can decrease interdependence among modules and bring convenience to developers. On the other hand, it can achieve code reuse.

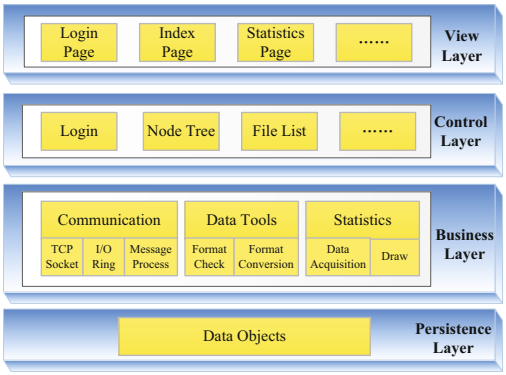


Fig. 5. Architecture of RMC

4 Evaluation

In this section, we compare BIDS with the model that does not perform the "bridgehead" mechanism. But in other aspects, it has similar collaborative sharings as BIDS does. We call it *Baseline*. We use provision time and WAN bandwidth consumption as performance metrics. Meanwhile, the influence of delay variation on BIDS's performance is also evaluated.

In order to better exhibit the BIDS's performance, we construct a simplified cloud data center network with a small amount of physical hosts. Within each IDC, we assign 200Mbps for intra-IDC links. All the physical hosts are connected via switches. For Inter-IDC, we construct diverse WAN environment by adjusting bandwidth size and delay time. The testing VM image can vary from 450MB to 4GB. Considering that the testing network size is small, it's improper to assign WAN bandwidth in the order of Gbps as real cloud data center does. Thus, we limit the bandwidth value to a small range.

4.1 Provision Time

In this part, we set up four IDCs act as "provider". About eight hosts in another IDC request image data in a random order. The WAN delay is set to 15ms.

For WAN bandwidth, some typical WAN types, like CAT-3 cable, CAT-4 cable, E-3 Europe and others, are emulated in the test. We repeated test, while varying the size of VM images:450MB, 1.6GB and 4GB. We record time each host takes, the final average results are shown in Fig. 6.

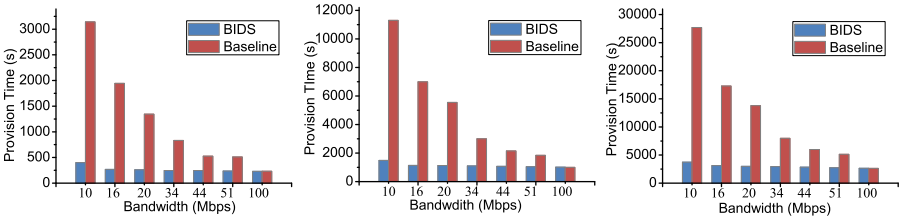


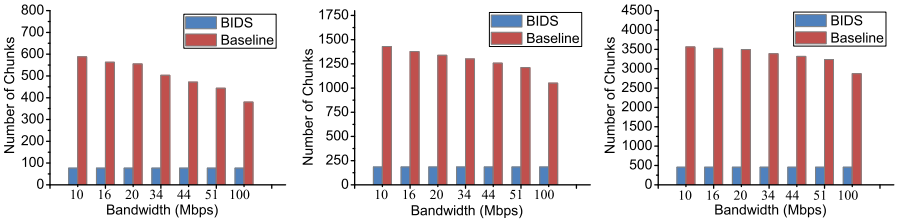
Fig. 6. Provision time for VM images over time using *Baseline*

Here, we make the following observations. First, the provision times both decrease as the WAN bandwidth increases. But the rate of decline in BIDS is lower. The reason is that *Baseline*'s bandwidth variation will affect all the requesters. But in BIDS, it only affects largely on DB nodes and has little influences on the requesters. Second, when the bandwidth is low, the performance gain of BIDS is significant, as much as 7x speedup. That's because BIDS fully utilizes the advantage of locality and high-speed LAN makes the image data quickly distributed to each host. But in *Baseline*, requesters have to put up with the low efficient of WAN. Third, when the bandwidth increases to certain degree, the decline rate in BIDS become quietly low. The reasonable explanation is that as bandwidth increases, the key factor of affecting performance in BIDS has changed from bandwidth to DB's distribution efficiency. But we have to say that BIDS is still quiet efficient, especially when the available bandwidth is low. Note that we can improve BIDS's performance manyfold by employing additional DBs.

4.2 Bandwidth Consumption

An efficient distribution model should keep low bandwidth consumption while improving distribution performance. In BIDS, we achieve it by decreasing the repetitive data flows of cross-IDCs. In this subsection, we analysis the bandwidth consumption of the two distribution models. Considering that each image file is divided into fixed-length chunks for distribution, we evaluate the bandwidth consumption through the statistical number of image chunks which are transferred via WAN. In this part, we employ the same test environment as Sect. 4.1 does. We conduct multiple tests, Fig. 7 shows the final statistical results.

It's obvious that BIDS has a fixed bandwidth consumption which comes from DB. Because just single DB node is employed for image distribution in our



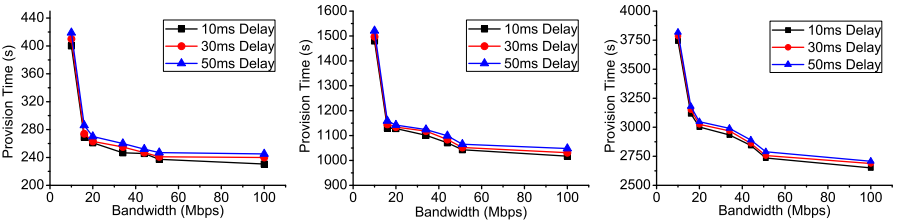
(a) 450MB (b) 1.6GB (c) 4GB

Fig. 7. WAN bandwidth consumption under different image file size

tests, the consumption value is changeless, but the bandwidth consumption for *Baseline* is quiet high. Considering that the image data sharing within IDC is faster than inter-IDC transmission, the request hosts within same IDC will own the same image data after some time running. Then they must have to require the chunks from other IDCs via WAN, which leads to high bandwidth consumption. In another aspect, we can see that the WAN bandwidth consumption reduces gradually as the bandwidth increases. The reason is that higher bandwidth makes it more possible for hosts to obtain chunks from the hosts within the same IDC.

4.3 Delay

In this subsection, we evaluate the influences of system performance when different delay time is assigned to WAN. Similar to Sect. 4.1, the provision time is employed as evaluation metric. Three typical delay times are adopted in our test:10ms, 30ms, 50ms. The testing results are shown in Fig. 8.



(a) 450MB (b) 1.6GB (c) 4GB

Fig. 8. Influence of delay variation on BIDS's provision time

As you can see from Fig. 8, delay variation doesn't bring about remarkable changes on provision time. The reason is as follows. In BIDS, chunk data is transferred in a pipelined manner. The receiver does not have to return acknowledge

messages until a complete chunk is received. Then MD5 checksums of chunks are used to ensure the correctness of transmission. We know that delay acts on distribution only when plenty of message exchange happen among hosts. But in BIDS, message exchange occurs only when the host needs to request new chunks, or retransmit error received chunks, or send checksums. But these only make up a small part of total distribution. Therefore, delay has limited influence on system performance.

5 Related Work

Nowadays, improving distribution efficiency of VM instances has great influence on the overall system performance. Therefore, many efforts have been made on this topic and lots of novel solutions are proposed.Among these researches, Schmidt et al.[8] discussed several distribution methods, including unicast, multicast, BitTorrent-like distribution. Wartel et al.[9] also proposed BitTorrent-like distribution model. In their model, they treat an entire VM image file as a BitTorrent seed file. Then Chowdhury et al.[10] revised the BitTorrent protocol and presented an architecture called Orchestra that controls both inflow and outflow data transmission to optimize performance. BJorkqvist et al[11] proposed a towtier network topology ignoring different network connections among edge nodes and this solution can reduce the retrieval latency for data centers. Different from the above distribution mechanisms, Peng et al.[5] proposed a more efficient sharing mechanism via utilizing common chunks in different VM images. Zhu et al[12] designed a new distribution mechanism called Twinkle. Twinkle reduces provisioning time by speeding up the initialization of VM instances using demand predication and partial page lunch. Epstein et al[13] focused on data placement on centralized storage servers. They tried to minimize the provision time via the optimization of staging schedules.

6 Conclusions

In cloud data centers, the low bandwidth and long delay of WAN is an important factor of affecting distribution. In this paper, we propose bridgehead mode to minimize the repetitive data flows of WAN while speeding up the image file distribution. Meanwhile, we design the version based collaborative sharing mechanism to speed up the image distribution and RMC make the administrators easy to manage all the image files. Final tests show that our system is efficient.

Acknowledgements. This work is supported by the National Science Foundation of China under Grant No.61340031,61073076,and 61202425, Ph.D. Programs Foundation of Ministry of Education of China under Grant No.20121102110018, and Key Technology R&D Program of Hebei Province, China under Grant No. 13200326D.

References

1. Chen, Z., Zhao, Y., Miao, X., Chen, Y.: Rapid provisioning of cloud infrastructure leveraging peer-to-peer networks. In: Proceeding of 29th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS), Washington, DC, pp. 22–26 (2009)
2. Chowdhury, M., Zaharia, M., Ma, J., Jordan, M.I.: Managing data transfers in computer clusters with orchestra. In: Proceeding of the ACM SIGCOMM 2011 Conference (SIGCOMM 2011), New York, pp. 98–109 (2011)
3. Reimer, D., Thomas, A., Ammons, G., Mummert, T.: Open black boxes: using semantic information to combat virtual machine image sprawl. In: Proceeding of the 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE), New York, pp. 111–120 (2008)
4. Jin, K., Miller, E.L.: The effectiveness of deduplication on virtual machine disk images. In: Proceeding of the Israeli Experimental Systems Conference (SYSTOR), New York, vol. (7) (2009)
5. Peng, C., Kim, M., Zhang, Z., Lei, H.: VDN: Virtual Machine Image Distribution Network for Cloud Data Centers. In: Proceeding of IEEE INFOCOM (INFOCOM 2012), Orlando, pp. 181–189 (2012)
6. Satyanarayanan, M., Richter, W., Ammons, G., Harkes, J.: The case for content search of vm clouds. In: IEEE 34th Computer Software and Applications Conference Workshops (COMPSACW), Seoul, pp. 382–387 (2010)
7. RHEV: Red Hat Enterprise Virtualization, `http://www.redhat.com/products/cloud-computing/virtualization`
8. Schmidt, M., Fallenbeck, N., Smith, M., Freisleben, B.: Efficient distribution of virtual machines for cloud computing. In: 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Pisa, pp. 567–574 (2010)
9. Wartel, R., Cass, T., Moreira, B., Roche, E.: Image distribution mechanisms in large scale cloud providers. In: 2th IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Indianapolis, pp. 112–117 (2010)
10. Chowdhury, M., Zaharia, M., Ma, J., Jordan, M.I.: Managing data transfers in computer clusters with orchestra. In: Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM 2011), New York, pp. 98–109 (2011)
11. Bjorkqvist, M., Chen, L.Y., Zhang, X.: Minimizing retrieval latency for content cloud. In: Proceedings of the IEEE INFOCOM (INFOCOM 2011), Shanghai, pp. 1080–1088 (2011)
12. Zhu, J., Jiang, Z., Xiao, Z.: Twinkle: A fast resource provisioning mechanism for internet services. In: Proceedings of the IEEE INFOCOM (INFOCOM 2011), Shanghai, pp. 802–810 (2011)
13. Epstein, A., Lorenz, D.H., Silvera, E., Shapira, I.: Virtual appliance content distribution for a global infrastructure cloud service. In: Proceedings of IEEE INFOCOM(INFOCOM 2010), San Diego, pp. 1–9 (2010)