Abstract—Network-booting is widely adopted in universities that have to maintain many client computers. In conventional network-booting systems, the primary bottleneck is the disk image distribution servers and the network to these servers. To eliminate this bottleneck, peer-to-peer (P2P) methods must work. However, existing P2P methods, including BitTorrent, do not work well for network-booting because they are highly optimized for distributing an entire large file, while network-booting requires certain parts of a large file. In the present study, aiming to solve the problems described above, we describe a new network-booting system that uses a P2P method. In our P2P-based network-booting system, a client node receives disk blocks not only from central distribution servers but also from the other client nodes that already have the demanded disk blocks. To the best of our knowledge, our network-booting system is the first effective implementation of a network-booting system that uses a P2P method in a local area network. Unlike conventional P2P systems, the proposed network-booting system can deal with demanded parts of a large disk image. We performed experiments with 112 client nodes in real classrooms on a university campus. The results of the experiments show that our implementation scales well as the number of client nodes grows.

Keywords—network-booting; P2P; Virtual machines

I. INTRODUCTION

Network-booting is widely adopted in companies and universities that have to maintain many client computers. Representative products of network-booting are Citrix Provisioning Services (PVS) and Apple NetBoot[5][4]. Network File System (NFS) rooted Linux is a free network-booting system for Linux[8]. A typical network-booting system consists of client computers, disk image distribution servers, and a local area network (LAN). In network-booting, administrators install applications and apply security patches to a single master computer, make a disk image from the master computer, and deploy the disk image to distribution servers. When client computers boot, they send requests to distribution servers over a LAN and receive disk blocks of the disk image. Network-booting has several advantages. First, it reduces administrative tasks including installing applications and applying patches. Second, rebooting a computer can revert any potentially dangerous modifications to files in client computers. For example, even if a user installs a malicious program in a computer, the next user can revert it by simply rebooting the computer.

Conventionally, the primary bottleneck is the distribution servers and the network to these servers. This bottleneck sometimes causes a problem in university campuses. When a class begins, students arrive at a classroom and turn on their computers almost at the same time. This simultaneous booting can delay a class. Since universities must provide classes for set periods of time, they must incur great expense for critical points in network-booting systems. For example, the authors’ university pays for 20 distribution servers to serve 985 client computers. It also pays for high speed campus networks (10 Gbps) between scattered classrooms and a central server room.

To eliminate the above-described bottleneck, peer-to-peer (P2P) methods must work. Fortunately, client computers in a classroom usually use the same disk image. If client computers can get the required disk blocks not only from distribution servers but also from other client computers, the load of distribution servers would be decreased, and traffic of the network would be reduced. As a result, a university would not have to pay considerable expense for distribution servers and the network.

However, existing P2P methods, including BitTorrent, do not work well for network-booting because they are highly optimized for distributing an entire large file, while network-booting requires certain parts of a large file. In the case of network-booting, some parts of a large file are more important than the other parts for fast booting. For example, when Microsoft Windows boots, it requires disk blocks that contains kernel programs, device drivers, and some initialization system programs in a certain order. Without such important disk blocks, client computers are blocked and users have to wait a long time.

In this study, we propose a network-booting system that uses a P2P method. To the best of our knowledge, our system is the first effective implementation of a network-booting system that uses a P2P method in a local area network. The system consists of central distribution servers and client nodes. A client node receives disk blocks not only from central distribution servers but also from the other client nodes that already have the demanded disk blocks. The central distribution servers also act as trackers of disk blocks as in BitTorrent. Unlike conventional P2P systems, our network-booting system can
deal with demanded parts of a large disk image. A client computer runs a hosted virtual machine monitor (VMM) to run unmodified guest operating systems, including Microsoft Windows. In a client, the proposed system captures disk block accesses from a virtual machine, and sends requests to other client nodes or central distribution servers.

We performed experiments with 112 client computers in real classrooms on a university campus. The results of the experiments show that our implementation scales well as the number of client nodes grows.

II. RELATED WORK

A. P2P file sharing systems

Many P2P file-sharing systems have been implemented in the practical world and the academic world [6][14][13][10]. For example, BitTorrent[7] is a P2P file-sharing protocol designed for distributing large files. BitTorrent divides a file into small blocks and distributes them. Clients can download those blocks from nodes that have already downloaded them. A central server, called a tracker, provides information about which nodes have the blocks.

O’Donnell et al. proposed a method that utilizes BitTorrent for distributing a disk image of a virtual machine in campus computer rooms[12]. In this method, blocks of a disk image are distributed in idle time, such as around midnight, and they are saved in local disks of client computers. Unlike IP multicast, BitTorrent works well for retransmission of error blocks.

As discussed in Section I, existing P2P file-sharing systems are highly optimized for distributing an entire large file, while network-booting requires certain parts of a large file. In [12], BitTorrent is used for distributing the entire disk image. Unlike that study, the present study focuses on network-booting that requires some parts of a large file.

B. Distributed file systems and disk image servers

Ceph[16] is a distributed file system that uses the following three kinds of servers to achieve high reliability and large capacity: a monitor for managing the system, a meta data server for managing meta data, and an object storage device for storing data as objects. A reliable, autonomic distributed object store (RADOS) is a distributed data store system, used as the backend of Ceph. A RADOS block device (RBD)[3] can handle a RADOS as a block device on Linux. A virtual-machine monitor KVM[9] can also utilize a RBD.

Sheepdog[11] is a distributed disk image service for KVM. In Sheepdog, virtual machine’s disk images are stored on multiple sheepdog nodes with some replicas on the basis of a consistent hashing algorithm. Since Sheepdog maintains at least two replicas on different sheepdog nodes, its high reliability is realized. The capacity of a sheepdog system can be increased by adding nodes.

In these distributed file systems and disk image servers, a small number of replica are used for reliability. A P2P method is not used for sharing disk blocks among clients. On the other hand, our network-booting system uses a P2P method to share disk images among clients.

C. The OS Circular project

The OS Circular project[15] allows users to boot various operating systems without installation as KNOPPIX. Unlike KNOPPIX that uses CD-ROMs and DVD-ROMs, this project uses remote disk image servers on the Internet. While an early system tried a P2P method over the Internet, it did not work well because of large network latency. A latter system used servers connected with a contents delivery network (CDN). In this study, we demonstrate an effective P2P network-booting system in a local area network.

III. P2P DISK IMAGE DISTRIBUTION FOR NETWORK-BOOTING

Our network-booting system targets the following environments:

1) Administrators manage many client computers in classrooms on a campus. They can install applications and apply security patches to a single master computer.
2) Disk images of client computers are accessed over a campus LAN every time these client computers boot up. A user can boot up a client computer and get a safe and stable OS image.
3) Client computers must run unmodified operating systems, including Microsoft Windows.

A. Problems and requirements

Existing network-booting systems, such as NFS rooted Linux[8], Citrix Provisioning Services (PVS)[5], and Apple NetBoot[4], require strong disk image distribution servers and the network to these servers. We have implemented a network-booting system that does not require such strong disk image distribution servers and the network. It has the following features.

1) P2P disk block sharing: Client nodes can receive disk blocks from other client nodes that already have demanded disk blocks.
2) Local caching: Client nodes can cache disk blocks in memory and local disks in order to improve performance.
3) Write isolation: When an OS writes disk blocks, these modifications are effective for the OS instance. Other OS instances running on other nodes cannot see the modifications.
4) Load balancing: If multiple nodes have the same disk block, they equally serve other client nodes.

B. Overview

Fig. 1 outlines our network-booting system. The system consists of central servers and clusters of client computers. The central servers store and distribute disk images. They are similar to those used in Citrix PVS and Apple NetBoot. In addition, the central servers help load balancing in a similar
manner to trackers in BitTorrent. Central servers are installed in a server room of a computer center.

Clusters of client computers are located in classrooms and used by users. As shown in Fig. 2, a client computer runs two operating systems: a host operating system running a VMM and P2P disk image provider, a program of the proposed P2P network-booting system; and a guest operating system running users’ applications. The latter operating system is called a UserEnvironment. Our P2P disk image provider provides disk blocks for the UserEnvironment.

When a user turns on a client computer, first, the host OS boots up from a local disk. Next, the host OS shows the user a list of bootable OSs. When the user chooses a single OS, the P2P disk image provider creates a virtual machine (VM) with the VMM, and runs a guest OS. The P2P disk image provider gets a cluster node list from a server. A cluster node list is a list of client nodes that are active in the same cluster. The P2P disk image provider chooses certain nodes from the list, registers them to its neighbor list, and sends a request for a blockmap. A blockmap is a list of nodes that contain pieces of meta information about disk blocks.

If the guest OS accesses a disk block, this access is captured by the VMM and the P2P disk image provider, which sends a disk block request to a neighboring node or central server according to the blockmap. The neighboring node or central server sends a disk block request to the P2P disk image provider, which sends the block to the guest OS. The P2P disk image provider periodically updates the blockmap.

When the user shuts down the UserEnvironment, the P2P disk image provider tells the central servers shutting down of the OS. The central servers then update the cluster node list and notify client nodes.

C. Neighbor discovery

In the case of a P2P method, a member node should choose near nodes (hereinafter referred to as neighbors) from the viewpoint of system performance. In our P2P network-booting system, a client node automatically chooses neighbors from the other client nodes connected to the same switch. As a client node uses more neighbors to get disk blocks than that the central servers use, the load on the central servers is decreased. Network traffic to the central servers is also reduced.

IV. IMPLEMENTATION

We have implemented the proposed network-booting system satisfies the requirements described in Section III in a cluster of client computers.

In the case of conventional network-booting systems, first, BIOS in ROM loads a boot loader and an OS kernel. Next, the OS kernel accesses the root filesystem through a network protocol or a thin abstraction layer of the network-booting system. For example, NFS-rooted Linux and Mac OS X use NFS to access the root file system in remote servers. Windows running in Citrix PVS uses a thin disk-abstraction layer that translates local disk accesses to network requests.

In the case of our network-booting system, first, BIOS in ROM boots a base host Linux from a local disk, and the host Linux runs the P2P disk image provider. An advantage of using a host Linux is that we can implement the P2P disk image provider as a regular application program running in Linux.

The P2P disk image provider runs KVM and creates a VM to run a guest OS. We can execute any OSs including unmodified Windows, that run in KVM. Since KVM translates the disk accesses of a guest OS into file accesses, a guest OS needs neither a network protocol nor an abstraction layer.

To capture file accesses, we use Filesystem in Userspace (FUSE) and fusefs[1]. Advantages of using FUSE are that complex kernel programming can be avoided, and productive script languages can be used. We used the programming language Ruby for developing the P2P disk image provider running on a client node. We also used Ruby for developing central servers.

A. Protocol

We used MessagePack RPC[2] for exchanging blocks of disk images and meta data. It was chosen because its performance is better than some protocols over HTTP. OS Circular took the HTTP for distributing a block of disk images because they require a regular CDN. On the other hand, our network-booting system does not use a CDN. The overhead of HTTP
communications could thus be reduced by using our a specific protocol over MessagePack RPC, which includes the following procedures for exchanging disk blocks and meta data.

```python
server.client_hello(os)
A client notifies a server that a user wishes to boot a specified OS. The server returns an active node list.
server.client_bye(os)
A client notifies a server that the user shutdown the OS.
callee.neighbor_hello(caller)
A client notifies another client node that the callee node is registered as the caller node’s neighbor.
server_or_neighbor.get_block(number)
A client sends a request for a specified disk block to a server or neighboring node. The server or neighboring node then returns the requested disk block.
callee.blockmap_request()
A client requests blockmap information from a neighboring node. The neighboring node then returns the blockmap to the client.
```

### B. Active client management in servers

Using the procedures `client_hello()` and `client_bye()`, the central servers maintain the active node list. Each entry in the active node list is a tuple of `<client ID, OS ID>`. When a server receives a request by the procedure `client_hello()`, the server adds the entry `<client ID, OS ID>` to the active node list and returns the current active node list to the client. The client then receives the active node list and chooses neighboring nodes from the active node list. When the server receives a request with the procedure `client_bye()`, the server removes the entry `<client ID, OS ID>` from the active node list.

### C. Blockmap management in a client

To update its blockmap, a client node sends a request to neighboring nodes with the procedure `blockmap_request()`. This procedure returns a blockmap, a list of client IDs that have the disk block. Using these results, the client maintains its own blockmap. Each entry of the blockmap is a tuple of `<Block Number, List of client ID>`. The client updates this blockmap periodically.

When the guest OS accesses a disk block, the client first looks up the block in the cache. If the block is found, the client returns the block. If the block is not found, the client searches the blockmap for the requested block number. If one or more entries are found in the blockmap, the client randomly chooses a single entry, extracts the client ID, and sends a request to the client with the procedure `get_block()`. If no entry is found in the blockmap, the client sends a request to a central server with the procedure `get_block()`.

---

**D. Replication of central servers**

To achieve high performance and high availability, it is generally desired to run multiple central servers. Our network-booting system allows multiple hot central servers to be run at the same time. A client node randomly chooses one of the central servers. Moreover, hot central servers periodically exchange active node lists with one another.

**V. Evaluation**

In this section, we show experimental results and the scalability of our network-booting system. In particular, we compare the network-traffic characteristics of the system with those of a simple NFS-based network-booting system. The network-traffic characteristics of the system were compared with those of a simple NFS-based network-booting system. The comparison results show that our network-booting system scaled well as the number of client nodes grew.

**A. Experimental environment**

An overview of the experimental environment is shown in Fig. 3. Experiments were performed in real computer rooms, at the University of Tsukuba, where 112 client nodes were installed. Each computer had a Core2Duo E8500 dual-core processor, 4 GB of memory, an 80 GB SerialATA disk, and a 1000Base-T network card. All client computers were connected to stacked Cisco Catalyst 3750E switches with 4 Gbps backplane capacity. All client nodes belong to the same cluster.

The computer used to run the central server had a Xeon X3440 quad-core processor, 8 GB of memory, 600 GB SAS RAID array, and a 1000Base-T network card. NFS servers and the central server of the proposed network-booting system were run on this computer. This computer was connected to the same network switch (mentioned above). We measured the network traffic characteristics of this computer through the `/proc/net/dev` file in the Linux proc file system.

Each client computer ran Linux CentOS 5.6 as a host OS and KVM 83.224.el5 as a virtual-machine monitor. In each client computer, a virtual machine was created, and a UserEnvironment was booted up. The target UserEnvironment used in the experiments was Microsoft Windows XP Professional.

The OS is stored in a disk image with the QCoW format of KVM (where QCoW stands for QEMU copy on write). If an OS writes data to a block in a client node, the original block is kept, and the data is appended to the disk image file in...
B. Scalability

We evaluated the scalability of our P2P network-booting system with increasing number of nodes. In particular, we compared the boot times of our system with those of a simple NFS-based network-booting system.

In NFS measurement, we enabled QCoW2’s differential disk image feature. In the case of this feature, the base image is read-only, and data is written in a differential file. In NFS measurement, the base image in the NFS server was shared among nodes, and a differential file was allocated on the node’s local disk.

In this study, we assumed the “boot start” of the UserEnvironment means the time that the P2P disk image provider executed the command to create a virtual machine.

In the case of Windows XP, automatic login was enabled, and Internet Explorer was set as the startup program. When Internet Explorer was started up, it opened a home-page and a CGI program was executed. We recorded the time that this CGI program was executed as the “boot finish time.” The time elapsed from the boot start time to the boot finish time is called BootingTime.

We measured the boot start times and boot finish times of both NFS and the proposed P2P systems after the disk cache was cleaned. Additionally, we cleaned the local cache of our P2P booting system before each measurement.

The measurements were performed with 15, 30, 61, and 112 nodes in both NFS and P2P cases. The nodes started to boot with a delay determined by a normal distribution with a time range of 60 seconds. Additionally, the corresponding times of our P2P booting system with block size parameters as 128KB, 256KB, 512KB, and 1024KB were measured.

All measurements were performed three times each. The three sets of results were sorted in order of median values, and the second set was used.

The relation between the number of nodes and block size is shown in Fig. 5, where the x-axis shows the number of nodes, and the y-axis shows BootingTime in seconds. As for the results for the NFS system, BootingTime increases linearly with increasing number of nodes. On the other hand, in the P2P cases, BootingTimes also increases, but the slopes of their curves are smaller than that of the NFS case in regard to number of nodes. These results show that our network-booting system scales well as the number of client nodes grows.

In terms of block size in the P2P cases, the 512KB and 1024KB cases had longer BootingTimes than the other cases. The block size influences the utilization ratio, which is the ratio of required disk image parts to the actually fetched parts. Decreasing block size improves the utilization ratio, but increases the overhead of RPC because the number of RPCs increases. Increasing the block size decreases the RPC overhead but decreases the utilization ratio.

In these experiments, the actual size of the required data was 174 MB. This number was taken from the disk access logs of FUSE. When the block size was 128 KB, the transferred size was 308 MB and utilization ratio was 56%. When the block size was 256 KB, it was 446 MB and the utilization ratio was 40%. When the block size was 512 KB, it was 717 MB and the utilization ratio was 24%. When the block size was 1024 KB, it was 1258 MB and the utilization ratio was 14%.

In regard to these experiments, it was supposed that the boot timing in a computer classroom follows a normal distribution. For example, at the beginning of the class, the peak of the boot timing is a few minutes before of the class starts. Students who take that class start their computers within a certain range of time (TimeRange). Calculated and actual boot-timing data concerning a computer classroom at the university are plotted in Fig. 4. This figure illustrates that the actual boot timing follows a normal distribution.
C. Network traffic characteristics

The network traffic on the central server (with 30 and 61 nodes) is shown in Fig. 6. The x-axis shows the elapsed time in seconds, and the y-axis shows the transfer rate in mega bits per second. The horizontal lines in Fig. 6 mean the duration of booting procedures. Each line starts at the boot start time and ends at the boot finish time. Fig. 6(a) and 6(c) show the results for the NFS cases, where the traffic increases rapidly and stays at maximum capacity. Fig. 6(b) and 6(d) show the results for the P2P 128KB case, where the traffic was also high at first in transfer. However, in the P2P cases, when some faster nodes started serving other nodes, the traffic became lighter.

In the NFS cases, when the number of active (booting) nodes increased, the load on the central server also increased. This load increase causes serious network congestion. On the other hand, in the cases of our P2P booting system, when active nodes increased, the number of block providers also increased. This situation is equivalent to increasing the number of servers and thus prevents network congestion.

D. Changing boot timing

We measured the relation between boot timing and Booting-Time in the NFS case and three P2P cases by using 61 nodes. We changed the boot timing in four ways: simultaneously, according to a normal distribution with a time range of 15 seconds, according to a normal distribution with a time range of 30 seconds, and according to a normal distribution with a time range of 60 seconds. BootingTime was measured in the same ways explained in Section V-B.

The measured BootingTimes are plotted in Fig. 7. In this figure, the y-axis represents BootingTime in seconds. Each bar represents the NFS, P2P 128KB, P2P 256KB and P2P 512KB cases. In the simultaneous-timing cases, P2P 128KB was best in terms of BootingTime, and P2P 256KB and 512KB are worse than NFS. Our P2P method did not work well in these two cases because most client nodes sent block requests to the central server. When most client nodes booted simultaneously and sent the request client_hello() to the central server, these client nodes received the empty list of current active
nodes. The small block size (128KB) is better than the other (bigger) block sizes in terms of BootingTime because the simultaneous booting included some fluctuation. In the case of a smaller block size, more delayed client nodes could send block requests to other faster client nodes.

Inserting delays according to a normal distribution shortened the BootingTimes, because it is improved the probability of fetching blocks from other nodes.

VI. SUMMARY

We proposed a new network-booting system that uses a P2P method for distributing disk images. In conventional network-booting systems, the primary bottleneck is the disk image distribution servers and the network to these servers. We eliminated this bottleneck with the P2P method. In our network-booting system, a P2P client node runs a hosted virtual machine monitor to run unmodified guest operating systems, including Microsoft Windows. A client node receives disk blocks of a virtual machine not only from central distribution servers but also from the other client nodes that already have the requested disk blocks. Unlike existing P2P methods which are highly optimized for distributing an entire large file, the P2P method used by our network-booting system can handle certain requested parts of a large disk image.

We performed experiments on booting times with 112 client computers in real classrooms on a university campus. The results of the experiments show that our P2P network-booting system scales well as the number of client nodes grows. In particular, our system completed booting in 51 seconds, while a simple NFS-based booting system completed booting in 244 seconds.

Future work includes eliminating the overheads in Ruby and fusefs implementations, and finding better methods for sharing blockmap information between nodes.

REFERENCES