HTTP-FUSE Xenoppix

Kuniyasu Suzaki[†] Toshiki Yagi[†] Kengo Iijima[†]

Kenji Kitagawa[††] Shuichi Tashiro[†††]

National Institute of Advanced Industrial Science and Technology[†]
Alpha Systems Inc.[††]
Information-Technology Promotion Agency, Japan[†††]

{k.suzaki,yagi-toshiki,k-iijima}@aist.go.jp
kitagake@alpha.co.jp, tashiro@ipa.go.jp

Abstract

We developed "HTTP-FUSE Xenoppix" which boots Linux, Plan9, and NetBSD on Virtual Machine Monitor "Xen" with a small bootable (6.5MB) CD-ROM. The bootable CD-ROM includes boot loader, kernel, and miniroot only and most part of files are obtained via Internet with network loopback device `HTTP-FUSE CLOOP`. It is made from cloop (Compressed Loopback block device) and FUSE (Filesystem USErspace). HTTP-FUSE CLOOP can reconstruct a block device from many small block files of HTTP servers. In this paper we describe the detail of the implementation and its performance.

1 Introduction

We have studied boot methods which make easy-to-use OSes and applications with small change of PC. One solution is a CD-bootable OS, but it requires downloading a big file (approximately 700MB ISO image) and burning a CD-ROM. Furthermore it requires remaking the entire CD-ROM when a bit of data is updated. The other solution is a Virtual Machine which enables us to install many OSes and applications easily. However, that requires installing virtual machine software.

We have developed "Xenoppix" [1], which is a combination of CD/DVD bootable Linux "KNOPPIX" [2] and Virtual Machine Monitor "Xen" [3, 4]. Xenoppix boots Linux (KNOPPIX) as Host OS and NetBSD or Plan9 as Guest OS with a bootable DVD only. KNOPPIX is advanced in automatic device detection and driver integration. It prepares the Xen environment and Guest OSes don't need to worry about lack of device drivers. For example, Plan9 is an advanced OS but has few device drivers. Xenoppix enables us to try easily such a special OS.

Unfortunately Xenoppix is still a DVD-bootable OS. It has a drawback of update difficulty. We wanted to get rid of the root file system from the Xenoppix DVD and manage it on an Internet server. It is a kind of thin client, but it aims that anonymous users can use sev-

eral OSes if they have a PC and Internet connectivity. It also makes for easy maintenance of OSes and applications because they are updated on the server.

There are several ways to expose a root file system on the Internet. There are NFS4 [5], OpenAFS [6], SFS [7, 8], SFSRO [9], and SHFS [10] as network file systems, and iSCSI [11] as network block device. Unfortunately most of them require special server software and special ports which are closed by a firewall. They also aren't considered to be opened public for anonymous users and have security problems.

To solve the problem we proposed a network loopback device, HTTP-FUSE CLOOP. The loopback device consists of small split and compressed block files which are exposed on an HTTP server. The block files are downloaded by the loopback device driver when the relevant address is accessed. The downloaded block files are decompressed and mapped to loopback device. These block files can be saved at a local storage as a cache.

The mapping of block address to a block file is done by an indexing table. The file name of block files is MD5 value of its contents. The indexing table has a list of MD5 file names. When a block is updated, a new block file is created with a new MD5 file name and the indexing table is renewed. The old block files don't need to erase. We can easily rollback with an old indexing table and block files. MD5 file names are used to increase security. The downloaded files are validated by file names. Furthermore, some block regions which have the same contents are represented by a file and reduce the total volume of virtual block device. This idea is resemble to Venti [13] of Plan9. In a later section, we compare it to our method.

We made HTTP-FUSE Xenoppix which includes HTTP-FUSE CLOOP. The size of bootable CD is 6.5MB and boots KNOPPIX as Host OS and NetBSD and Plan9 as Guest OS on Xen. In this paper we describe the detail of HTTP-FUSE Xenoppix. In section 2 we introduce Xenoppix. The detail of network block device "HTTP-FUSE CLOOP" is described in Section 3. The current implementation of HTTP-FUSE Xenoppix is presented in Section 4 and its performance is reported in Section 5. We discuss related works and future plans in Section 6, and conclude in Section 7.

2 Xenoppix

In this section we describe the detail of Xenoppix. Xenoppix is a combination of KNOPPIX and Xen.

2.1 KNOPPIX

KNOPPIX [2] is a bootable CD/DVD Linux with a collection of GNU/Linux software. It is not necessary to install anything on a hard disk, and it enables running GNU/Linux on any PC. KNOPPIX can be used as a normal desktop Linux because it includes a powerful graphical desktop environment (KDE), office software (OpenOffice.org), Web browsers (Konqueror and Mozilla), image manipulation software (GIMP), many games, etc. CD-bootable Linux isn't an exclusive feature of KNOPPIX. There are many distributions: DemoLinux, Mepis, Slax, Adios, etc. Among them, KNOPPIX is a leading, popular CD bootable Linux, because its automatic hardware detection/configuration (AutoConfig) and compressed loopback device (cloop) are excellent.

2.1.1 AutoConfig

AutoConfig function of KNOPPIX detects individual devices and loads suitable device drivers. AutoConfig is achieved by

the `/etc/init.d/knoppix-autoconfig` script at boot time. The script consists of a hardware detection part and a driver setup part. Hardware detection is done by the `hwsetup` binary which is based on `kudzu` [12], the Red Hat Linux hardware probing library. After hardware detection, drivers are set up by setup-scripts like `mkxf86config`. If a network card is detected and DHCP is available, an IP address is automatically set up.

2.1.2 cloop

Cloop is a compressed loopback device which supports file system independence, transparent decompression, and read only mounts. It reduces the space needed on the CD to about 50% down to 25% of the original file system. KNOPPIX stores its root file system to a cloop file and mounts it at boot time. 700MB volume of CD-ROM is almost occupied by a cloop file `/KNOPPIX/KNOPPIX`. The rest of the volume is files for boot. Figure 1 shows the image of KNOPPIX CD-ROM. Cloop reduces access data of CD and make data-read fast with a help of on-the-fly decompression.

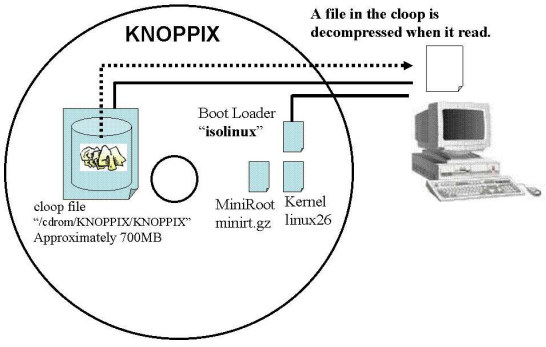


Figure 1: The contents of KNOPPIX

2.2 Xen

Xen [3, 4] is a virtual machine monitor (VMM) for x86 that supports execution of multiple Guest OSes with close-to-native performance and resource isolation.

Xen uses a very different technique than the traditional virtualization, namely *para-virtualization*. In paravirtualization, the Guest OS is ported to an idealized hardware layer which completely virtualizes all hardware interfaces. When the OS updates hardware data structures, such as the page table, or initiates a DMA operation, it makes calls into an API that is offered by VMM. The VMM keeps stack of all changes made by the OS and optimally decides how to modify the hardware on any context switch. VMM is mapped into the address space of each Guest OS, minimizing the context switch time between any OS and VMM. Finally, by co-operatively working with the Guest OSes, VMM gains additional insight into the intentions of the OS, and can make the OS aware of the fact that it has been virtualized. The para-virtualization is enabled by small patched to the Host and Guest OSes. On Xen-2.0.6, available Host OS is Linux and Guest OSes are Linux, Free BSD, NetBSD, and Plan9, which are all open source OSes.

2.3 Xenoppix = KNOPPIX + Xen

We customized KNOPPIX to include a virtual machine monitor Xen. We call it *Xenoppix*. Xenoppix sets up device drivers using Auto-Config function of KNOPPIX and enables to boot a Guest OS on Xen. The X Window System is prepared by KNOPPIX and the GUI of Guest OS is mapped to X Windows using VNC full-screen mode. It shows that Guest OS boots as a standalone OS. Furthermore the Guest OS can work as a server because it gets IP address from external DHCP with VIF-Bridge of Xen.

The update of files are covered by UNIONFS [14] on Host OS and Device Mapper [16] on Guest OS. UNIONFS is a stackable file system

Linux 2.6.12 (Domain0)	Xen VMM	0.12MB
	kernel with Xen patch	1.3MB
	miniroot	0.89MB
	Root File System	870MB
NetBSD (DomainU)	kernel with Xen patch	1.7MB
	Root File System	140MB
Plan9 (DomainU)	kernel with Xen patch	1.9MB
	Root File System	140MB

Table 1: Size of files in Xenoppix DVD (1.1GB)

which allows us CopyOnWrite on read only file system. Device mapper is a Linux kernel module for logical volume management. It enables us to CopyOnWrite on the device level.

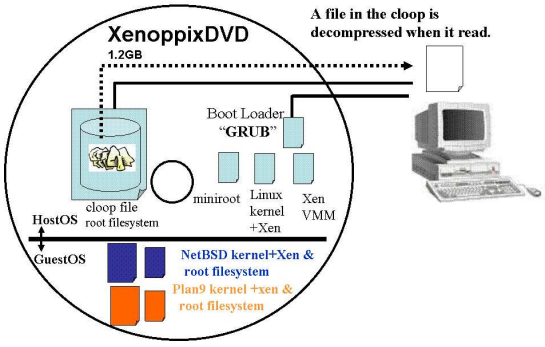


Figure 2: The contents of Xenoppix

Current Xenoppix includes 2 Guest OSes; NetBSD and Plan9. Figure 2 shows the contents of Xenoppix DVD which based on KNOPPIX 4.0.2 CD version and Xen 2.0.6. Table 1 shows the size of main files on Xenoppix DVD. The boot loader "isolinux" is replaced by "GRUB" because Xen requires loading VMM before Linux kernel. Linux kernel and miniroot is loaded as 2nd and 3rd modules by GRUB. The Linux kernel with Xen patch boot at first and prepare device drivers with AutoConfig of KNOPPIX. After that Guest OS is booted on Xen.

3 HTTP-FUSE CLOOP: A Network Loopback Device of split and compressed block files

We developed a network loopback device of split and compressed block files. It is based on a compressed loopback device "cloop" and a user-space file system "FUSE" [15]. So we call it HTTP-FUSE CLOOP.

3.1 cloop: Compressed Loopback Device

Cloop is a compressed loopback device which saves virtual block device in a file. A cloop file is made from a block device which has already included root file system on it (Figure 3). The block device is split by a fixed size (KNOPPIX's default is 64KB) and compressed by zlib. The data are saved in a cloop file with a header which is a index of compressed blocks.

CD KNOPPIX has a 700MB cloop file which stores 2GB block device. Cloop is a block device level abstraction and doesn't care about the file system. So any file systems can be saved to cloop file, for example iso, ext2, etc. We adapt the ext2 file system (block size of file system is 4KB) as default.

A cloop file is setup as a loopback device at `/dev/cloop*` and the file system is mounted (Figure 3). When a read request is issued from the file system, cloop driver read a relevant cloop block data from a cloop file using index header and decompresses the data at cloop driver's cache (64KB). Cloop driver returns request block unit (4KB) data of EXT2 from the cache. The cached date doesn't erase and is used when the next read request is fit to the cloop block.

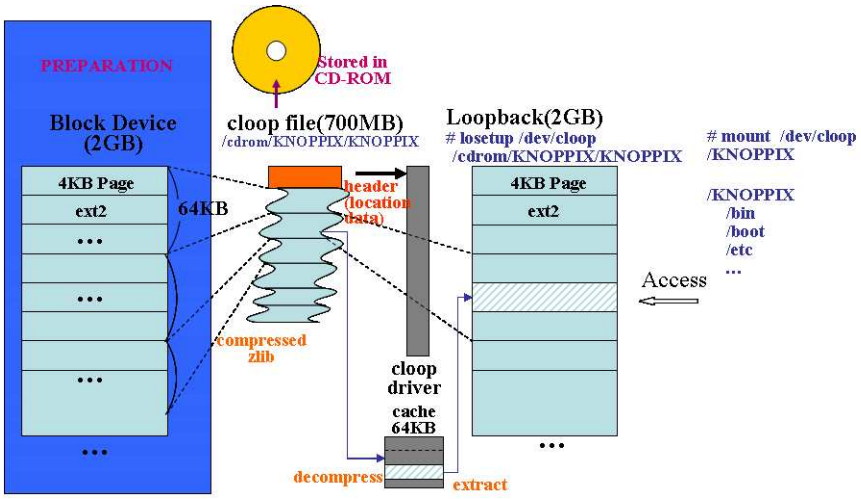


Figure 3: Cloop of KNOPPIX

3.2 Drawback and Improvement of Cloop

Coop is convenient because it saves block device to a file and makes small. However a cloop file itself becomes a big file. The size of traditional CD-KNOPPIX is about 700MB. It must be treated as one file and takes much time to download. Furthermore a big cloop file has to re-build when a bit of date is updated.

To solve this problem, we develop a new block device. Data of a block device is divided by a fixed block size and saved to many small block files. Saved data are also compressed. Block files are treated as network transparent between local and remote. So block files are location free. Local storage acts as a cache. The feature of the network loopback device follows:

- A block file is made of each 256KB block device. A block data is compressed by zlib and saved to a block file.
  - The block split size "64KB" is too small and makes too many files.
- Block files are mapped to a loopback device with `index.idx` file. `index.idx` acts a header of cloop file.
- The loopback device is a virtual device. The mapping of block file is done when a relevant read request is issued.
  - After mapping, the block file is erasable from local storage, because it can be re-downloaded from Internet.
- A name of block file is the hash value of MD5. If the block contents are same, they are held together a same name file and reduce total file size. The block contents become identifiable because it is confirmed by the MD5 file name.
- Block files are downloadable from HTTP server because HTTP is expected to be strong file delivery infrastructure. For examples, mirror servers and proxy servers.

We used virtual file system "FUSE" (File system in USEr-space) [15] to implement the virtual loopback device. This situation resembles to loopback device which is a virtual block device on a file system. The merit of virtualization is to make easy to treat low level device.

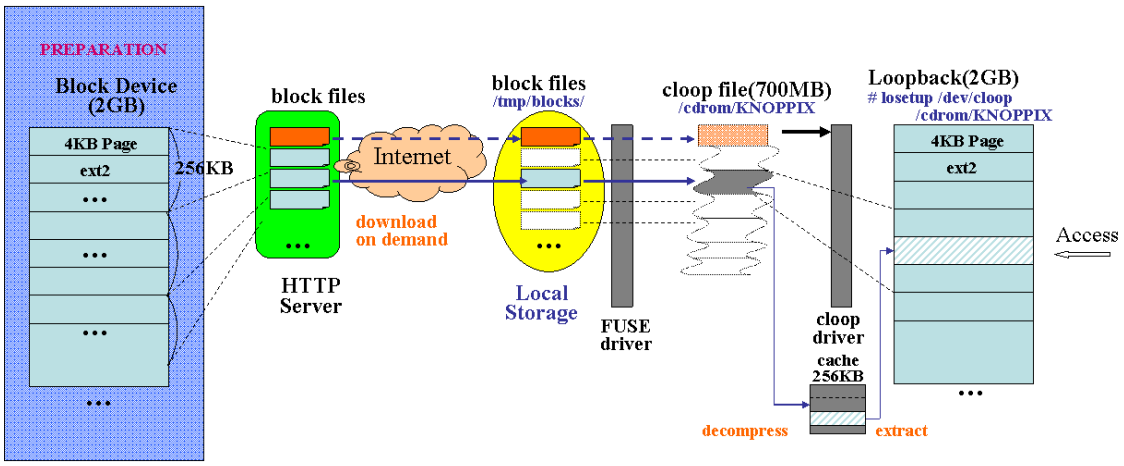


Figure 4: Structure of HTTP-FUSE CLOOP

Figure 4 shows structure of HTTP-FUSE CLOOP. The driver is implemented as a part of FUSE wrapper program. Block files and `index.idx` are also made from a block device which includes root file system. The block files and `index.idx` are downloadable by HTTP server.

`index.idx` file is downloaded at first because it is used to setup HTTP-FUSE CLOOP. When a read request is issued, HTTP-FUSE CLOOP driver searches a relevant block file with `index.idx` file. If a relevant file exists on a local storage, the file is used. If not, the file is downloaded from Internet. The download program is implemented by "libcurl" and is included in the FUSE wrapper. The downloaded block file is stored in RAM-Disk or local storage. If the storage space is not enough (more than 80% is used), the previous downloaded files are removed by LIFO of water mark algorithm.

3.3 Update by difference blocks

The addressing of HTTP-FUSE CLOOP is managed by the mapping table of `index.idx`. So the update of HTTP-FUSE CLOOP is done by adding updated block files and renewing `index.idx`. The rest block files are reusable. To achieve this function, the file system on HTTP-FUSE CLOOP have to treat block unit update as EXT2 file system. "iso9660" is not suitable because partial update of iso9660 changes the location of following blocks. The updated block is saved to a file with new file name of MD5. Collision of file name will be rarely happened. Even if a collision happens, we can check and fix before uploading the block files.

Figure 5 shows an example of update of HTTP-FUSE CLOOP. It is useful to update applications of KNOPPIX, especially for security update. Furthermore we can rollback to an old file system if old `index.idx` and block files exist.

4 Implementation of HTTP-FUSE Xenoppix

We adapt HTTP-FUSE CLOOP to Xenoppix. HTTP-FUSE CLOOP driver and setup software are included in a miniroot because they

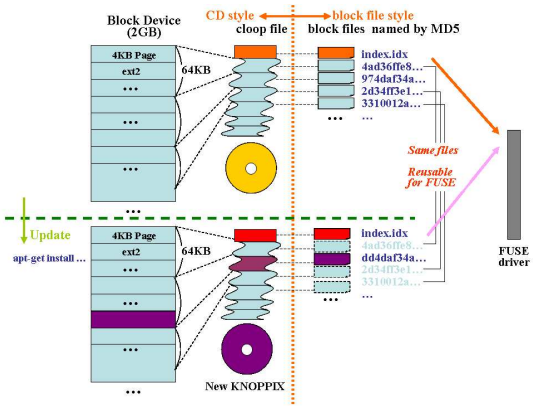


Figure 5: Update of HTTP-FUSE CLOOP

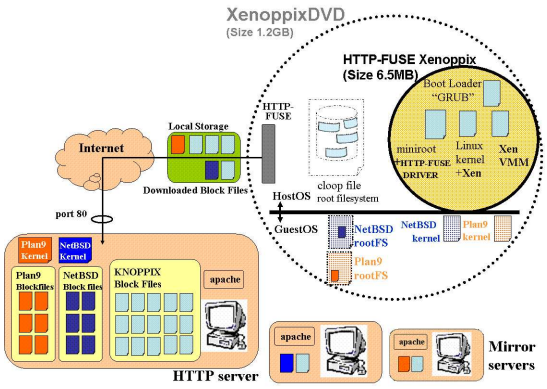


Figure 6: Usage model of HTTP-FUSE KNOPPIX

are used before mounting the root file system. The software to boot initial Host OS environment is stored in 6.5MB ISO image. The root file system of Host OS is downloaded via Internet with HTTP-FUSE CLOOP. The files for Guest OS are also downloaded via Internet on demand. Figure 6 shows the usage model of HTTP-FUSE Xenoppix.

The downloaded block files are saved at a local storage if it is available. The local storage works as a cache. If the all block files are saved to a local storage, HTTP-FUSE Xenoppix doesn't need to download anymore. So HTTP-FUSE Xenoppix can boot from local storage as well as HTTP server.

4.1 Drawback and Settlement

Access request is passed as the following steps on HTTP-FUSE CLOOP.

$$\text{ext2} \rightarrow \text{cloop} \rightarrow \text{FUSE} \rightarrow$$
$$(\text{HTTP Internet}) \rightarrow \text{block-file}$$

Cloop is a virtual block device and the access request is sequential. It means only one read request is issued to cloop. It turns to download a small block file. So HTTP-FUSE CLOOP is vulnerable to network latency and causes narrow band width. It can't make bandwidth extension with multi-connections, which is a technique used by NFS, because it can't accept multiple read requests. Especially boot time has no cue to hide latency and suffers affect of latency.

To solve this problem, we add two functions, `netselect` and `DLAHEAD`. Netselect enables us to find the best site and DLAHEAD enables us to download the necessary block files in advance.

4.1.1 netselect

`netselect` is a software that selects the shortest latency site among candidates by using `ping`. We prepare several HTTP sites for HTTP-FUSE Xenoppix and add a boot option of `netselect` to find nearest site automatically. Figure 7 shows the location of the sites. We arranged the sites to be dispersed across the globe as possible as we could. However the sites are centered in North America and Japan because of the cost to keep sites.

`netselect` is expected to make fast boot of

Figure 7: Web Sites for HTTP-FUSE Xnoppix

HTTP-FUSE Xenoppix. However it can't estimate the bandwidth and traffic congestion. So it doesn't always find the best site.

4.1.2 DLAHEAD

We develop a function named DLAHEAD (download ahead). DLAHEAD downloads the necessary block files in advance. The list of necessary block files is made from the boot profile of HTTP-FUSE Xenoppix. DLAHEAD establishes multiple connections and downloads block files in parallel. The default number of connections is four. The downloaded block files are saved to a local storage. The files work as a cache and omit download of HTTP-FUSE-CLOOP driver.

DLAHEAD is reasonable settlement but isn't almighty. It doesn't cover all read request. For examples, special boot options or unexpected device drivers. At that time, download is done by HTTP-FUSE-CLOOP driver and suffers network latency.

5 Performance Evaluation

We evaluated performance of HTTP-FUSE Xenoppix at boot time. We analyzed the effect of DLAHEAD and affect of network latency.

loopback file	Number of block files	Size of block file		Amount of files
HostOS (KNOPPIX) 680MB	7,483	Max Min Ave	262,230 277 94,740	6800 MB
GuestOS (NetBSD) 140MB	1,559	Max Min Ave	253,977 277 86,642	130 MB
GuestOS (Plan9) 140MB	1,346	Max Min Ave	262,230 277 73,161	94 MB

Table 2: Feature of block files (256KB split)

We prepared test environment for this evaluation. The server machine was Dell PowerEdge 1600SC with Pentium Xeon 2.8GHz, 1000M MIC and 4GB memory. It ran apache2 as a HTTP server. The client machine was IBM ThinkPAD T23 with Pentium III 1GHz, 100M NIC, 1GB memory, and 24x CD-ROM drive. To synthesize network latency we used "dummynet" of FreeBSD. We prepared FreeBSD machine which had 2 NICs and acted as a network bridge. The synthesized network latency was 100msec.

At first we analyzed the feature of block files of HTTP-FUSE Xenoppix. Table 2 shows the size of block files. DLAHEAD downloads 729 block files (62MB) with four extra HTTP connections.

5.1 Boot Time

The boot time was measured from prompt of "GRUB" to the end of GUI setup. KNOPPIX (Host OS) requires much time because the default desktop manager, KDE, is rich and needs many block files. NetBSD and Plan9 boot as a Guest OS on Xen. When a Guest OS is booted, the Host OS (KNOPPIX) prepares X Windows only. X Windows is used by full screen VNC of Guest OS. On the VNC, NetBSD boots until XDM (X Display Manager) and Pan9 boots until its login console.

Table 3 shows boot time of each OS of HTTP-FUSE Xenoppix. Each boot time includes

	Xenoppix DVD	No Latency	No Latency +DLAHEAD	100msec Latency	100msec Latency +DLAHEAD
KNOPPIX	184	173 94%	157 (16, 9%) 85%	432 235%	282 (150, 35%) 153%
NetBSD on Xen	162	176 108%	166 (10, 6%) 102%	384 237%	231 (153, 40%) 143%
Plan9 on Xen	127	135 106%	130 (5, 4%) 102%	340 268%	200 (140, 41%) 157%

Table 3: Boot Time (Sec). Upper part shows boot time and lower part shows percentage compared to boot time of Xenoppix DVD. The value in parenthesis shows time and percent shortened by DLAHEAD.

the time consumed to setup of HTTP-FUSE CLOOP. The setup took 44 seconds on IBM ThinkPAD T23. The table shows the time of DVD Xenoppix as a reference.

The boot time of HTTP-FUSE Xenoppix was almost same to DVD boot time when the network latency is not synthesized. This result means that HTTP-FUSE Xenoppix is valid at LAN environment, because HTTP-FUSE Xenoppix makes easy to maintenance. At this environment the effect of DLAHEAD was little. It was less than 10%.

However it became prominent at 100msec latency. It made 35%, 40%, and 41% faster than no DLAHEAD on KNOPPIX, NetBSD, and Plan9 respectively. The total boot time of each OS was more than 2 times of DVD Xenoppix when DLAHEAD is not enabled. But it became about 1.5 times when DLAHEAD is enabled. The results show necessity of DLAHEAD on Internet.

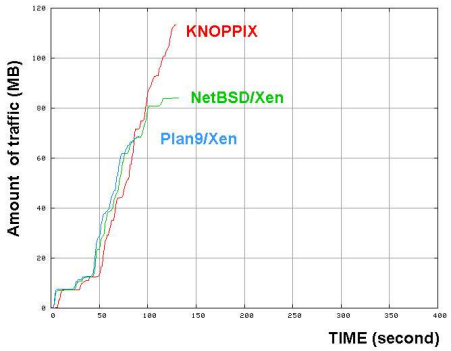
5.2 Trace of Traffic and Throughput

We made graphs of traffic and throughput at boot time. Figures 8 and 9 show the results. From these results we found the amounts of download were 110MB, 84MB and 69MB for KNOPPIX, NetBSD and Plan9 respectively, when DLAHEAD was not enabled. They were 115MB, 92MB, and 77MB when DLAHEAD was enabled. DLAHEAD increased the total amount of download because the block list was a general and included some unused block files.

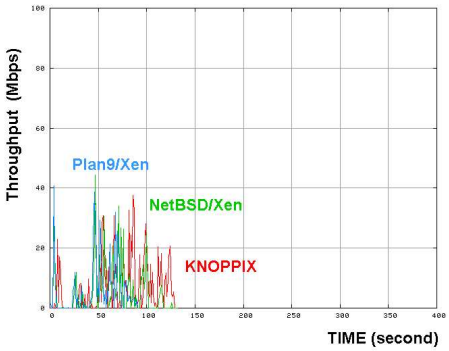
To compare Figure 8-A and 9-A we found the affect of network latency. This results show original HTTP-FUSE CLOOP was sensitive of network latency. The peak throughput was about 40 Mbps at no latency (Figure 8-C) but they became only 4Mpbs at 100msec latency (Figure 9-C). This results show importance to find the nearest download site.

To compare Figures 8-D and 9-D we found the effect of DLAHEAD. On each case the DLAHEAD makes throughput wide but the effect is different. At no latency the peak band throughput was about 100Mpbs. It was a maximum throughput at the environment and DLAHEAD finished at 10 second. The boot process doesn't catch up. So the total boot time is not shortened so much. This results show the advantage of DLAHEAD will be not recognized at LAN environment. However DLAHEAD shows the effect at 100msec latency. The peak throughput became 16Mbps (Figure 9-D). It was four times of the case of no DLAHEAD. The peak throughput continued till the end of DLAHEAD. At this time downloaded block files were took over to boot process and cut the affect of network latency.

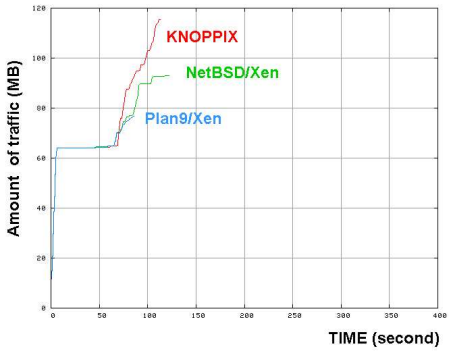
Unfortunately DLAHEAD is a kind of counter measure. After the use of downloaded block



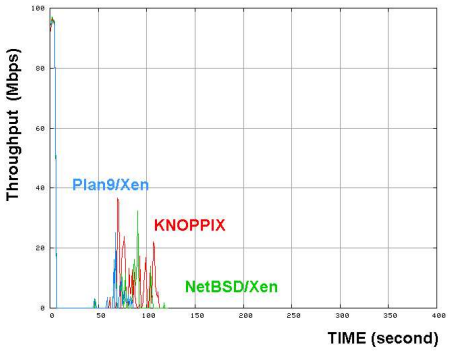
A. Traffic without DLAHEAD



B. Throughput without DLAHEAD

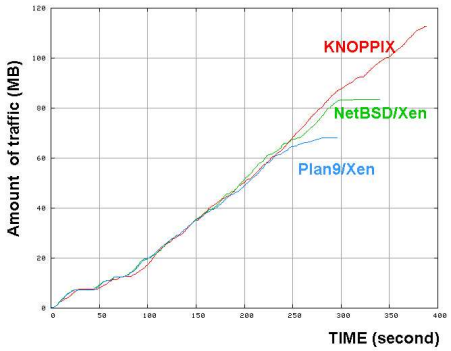


C. Traffic with DLAHEAD

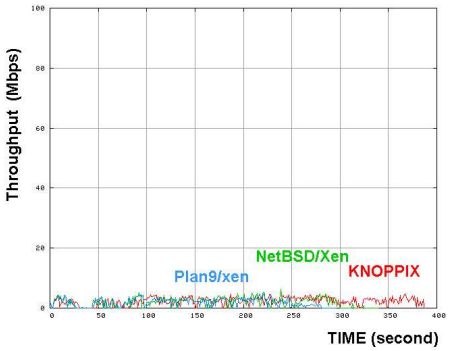


D. Throughput with DLAHEAD

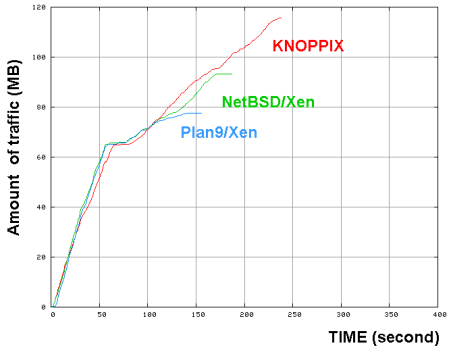
Figure 8: Network Traffic and Throughput of Boot of HTTP-FUSE Xenoppix at No Latency



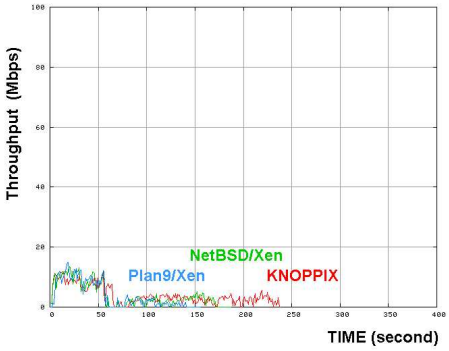
A. Traffic without DLAHEAD



B. Throughput without DLAHEAD



C. Traffic with DLAHEAD



D. Throughput with DLAHEAD

Figure 9: Network Traffic and Throughput of Boot of HTTP-FUSE Xenoppix at 100 msec Latency

files, the HTTP-FUSE driver has to download with a single connection. Network latency concerned to all download. So it is much important to prepare worldwide HTTP servers as candidates of netselect and find a short latency one.

6 Discussions

6.1 Venti of Plan9

"Venti" [13] is an archival block storage server for Plan9. In this system, a unique hash of a block's contents acts as the block identifier for read and write operations. This approach enforces a write-once policy, preventing accidental or malicious destruction of data. In addition, duplicate copies of a block can be coalesced, reducing the consumption of storage and simplifying the implementation of clients. Venti is a building block for constructing a variety of storage applications such as logical backup, physical backup, and snapshot file systems.

On HTTP-FUSE CLOOP, block data are also managed by a unique hash MD5. However each block data is saved as a file. File is a logical storage and easy to treat. It is easy to make copies of block files and distribute them. So we could use HTTP servers to distribute them via Internet.

In comparison with Venti, the overhead of HTTP-FUSE KNOPPIX seems to take much time, especially it uses user-space file system FUSE. However the most overhead was caused by downloading a block file on Internet. This point would be cared by netselect and DLA-HEAD partially. But it's not enough. We will make an improvement of download method. The native overhead of the driver is mentioned in following section.

6.2 Deployment of Virtual Machine for OS migration

There are several researches of deployment of virtual machine for OS migration. "soulPads" and "XenFS" have close relations to HTTP-FUSE Xenoppix.

SoulPads [17] is same concept as Xenoppix. It uses AutoConfig of KNOPPIX to prepare Host OS environment and VMware Workstation to run Guest OS. It is reasonable implementation but requires commercial license. Even if VMware Workstation is replaced with VMware Player, it is not re-distributed without permission. Furthermore SoulPads is based on potable disk device and doesn't have extension like a HTTP-FUSE Xenoppix of 6.5MB CD-ROM.

XenFS [18] is project sharing disk image of Xen. Unfortunately it is under development. According project plan, the implementation of XenFS is tightly coupling to API of Xen and aims high performance. The target is same to HTTP-FUSE Xenoppix but it uses a device level abstraction HTTP-FUSE CLOOP. So it doesn't concern to File System.

6.3 How to distribute block files

Current implementation used fixed HTTP servers to distribute block files. They were useful but have to be maintained all servers when block files are updated. So the cost to keep servers is not small. We want to distribute block files automatically with a help of P2P. The candidates are "coral" [19, 20] and "Dijjer" [21]. Unfortunately their current implementations are not good at keeping quick response and distributing many small files.

6.4 Trusted boot by TPM

The block files are confirmed its contents by the MD5 file names. However there is no warranty of index file. If a user gets a fraud index file, current HTTP-FUSE Xenoppix has no way to check. So, we want to integrate the trusted boot method offered TPM (Trusted Platform Module) chip [22]. It will help to upgrade security level.

6.5 Overhead of HTTP-FUSE CLOOP

The overhead of HTTP-FUSE CLOOP on Xenoppix doesn't look reasonable. It seems to be some affects from Linux kernel patch of Xen, because HTTP-FUSE CLOOP on normal Linux kernel showed better performance. Unfortunately we have not found the reason yet. We will analyze the behavior of HTTP-FUSE CLOOP driver and improve the performance.

7 Conclusions

We developed a network loopback device "HTTP-FUSE CLOOP" which is constructed with split-and-compressed block files from HTTP servers. We adopted it to Xenoppix and made HTTP-FUSE Xenoppix which enabled to boot Linux as a Guest OS and NetBSD and Plan9 as a Guest OS with 6.5MB boottalbe CD.

The boot time of HTTP-FUSE Xenoppix was almost same to original DVD Xenoppix at LAN environment. Unfortunately it became worse at Internet environment. It would be more than two times of original DVD boot time when the latency was 100msec. However we developed methods to improve performance, netselect and DLAHEAD. When DLAHEAD is enabled, the boot time was improved about 40%.

The implementation hasn't matured yet. To make good performance we have to improve the driver of HTTP-FUSE CLOOP as well as block file distribution methods. We also have to increase security. Furthermore we want to try dynamic OS migration using HTTP-FUSE Xenoppix in near future.

References

[1] Xenoppix, `http://unit.aist.go.jp/itri/knoppix/xen/index-en.html`

[2] KNOPPIX, `http://www.knopper.net/knoppix`

[3] Xen, `http://www.cl.cam.ac.uk/Research/SRG/netos/xen/`

[4] I. Pratt, *Xen 3.0 and the Art of Virtualization*, Ottawa Linux Symposium 2005

[5] NFS4, `http://www.nfsv4.org/`

[6] Open-AFS, `http://www.openafs.org/`

[7] SFS, `http://www.fs.net/sfswww`

[8] D. Maziéres, *Self-certifying file system, PhD thesis, MIT, 2000*

[9] K. Fu, M.F. Kaashoek, and D. Maziéres, *Fast and secure distributed read-only file system*, ACM Transactions on Computer Systems 20(1), 2002

[10] SHFS, `http://shfs.sourceforge.net/`

[11] iSCSI, `http://www.ietf.org/rfc/rfc3720.txt`

[12] kudzu; Red Hat Linux hardware probing library, http://rhlinux.redhat.com/kudzu/

[13] S. Quinlan and D. Dorward, *Venti: a new approach to archival storage*, USENIX Conference on File and Storage Technologies 2002

[14] UNIONFS, http://www.fsl.cs.sunysb.edu/project-unionfs.html

[15] FUSE, http://fuse.sourceforge.net/

[16] DeviceMapper, http://sources.redhat.com/dm/

[17] R. Cáceres, C. Carter, C. Narayanaswami, and M. Raghunath, *Reincarnating PCs with Portable SoulPads*, 3rd International Conference on Mobile Systems, Applications, and Services, 2005

[18] XenFS, http://wiki.xensource.com/xenwiki/XenFS

[19] Coral Project, http://www.coralcdn.org/

[20] M.J. Freedman, E. Freudenthal, and D. Maziéres, *Democratizing Content Publication with Coral*, USENIX Symposium on Networked Systems Design and Implementation 2004

[21] Dijjer Project, http://www.dijjer.org/

[22] TPM of Trusted Computing Group, https://www.trustedcomputinggroup.org/groups/tpm/

Proceedings of the Linux Symposium

Volume Two

July 19th–22nd, 2006
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc.*
C. Craig Ross, *Linux Symposium*

Review Committee

Jeff Garzik, *Red Hat Software*
Gerrit Huizenga, *IBM*
Dave Jones, *Red Hat Software*
Ben LaHaise, *Intel Corporation*
Matt Mackall, *Selenic Consulting*
Patrick Mochel, *Intel Corporation*
C. Craig Ross, *Linux Symposium*
Andrew Hutton, *Steamballoon, Inc.*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*
David M. Fellows, *Fellows and Carr, Inc.*
Kyle McMartin