unionfsを用いたディスクレスシステムの実装とその評価

桝田 秀夫†, 齊藤 明紀‡.
h-masuda@kit.ac.jp, saitoh@kankyo-u.ac.jp
† 京都工芸繊維大学 情報科学センター
‡ 鳥取環境大学 情報システム学科

概要 近年，Diskless構成をとるコンピュータシステムがTCOの削減やセキュリティの強化に適しているとして注目されている．しかし，各クライアントPCのローカルHDDの内容をそのままサーバ上に配置しネットワーク経由で使うだけでは，クライアントPCの台数分のディスクイメージをメンテナンスする必要が生じ，TCO削減などの効果が低くなってしまう．本稿では，Diskless構成をunionfsを利用し，基本的に単一のディスクイメージを使って複数のクライアントPCを稼働させるシステムの設計とその実装をおこなった．本方式は，NetBSD, Vine Linuxの各オペレーティングシステム上で動作することを確認した．また，unionfsを使った場合と使わない場合とで，起動時には10%程度の高速化の可能性があることを確認した．

キーワード ディスクレス, 単一イメージ, unionfs, Linux, NetBSD

Implementation of the diskless UNIX system using unionfs

Hideo Masuda[1] , Akinori Saitoh[2].
h-masuda@kit.ac.jp, saitoh@kankyo-u.ac.jp
[1] Center for Information Science, Kyoto Institute of Technology
[2] Department of Information System, Tottori University of Environmental Studies

Abstract The diskless computer system is suitable for reducing the total cost of ownership and protecting an information leak. However OS image of a client PC just migrates from local HDD to server's disk, it causes that many copies waste the server's disk and updating of OS image becomes costly.

In this paper, we implement the diskless computer system using unionfs operated by single OS image from all clients. The diskless mechanism using unionfs can be used by both Vine Linux and NetBSD. We also evaluate that unionfs can reduce a booting time about 10% faster.

keywords Diskless, Single image, unionfs, Linux, NetBSD

1 はじめに

大学の情報基盤センターや総合情報処理センターなど，大量の端末設備を有し，全学の学生が利用者であり，講義や演習を行う教室も提供するような計算機システムでは，可用性や安定性，セキュリティを保ちつつ，TCO(Total Cost of Ownership)を削減して運用を行わなければならない [1] .

本稿では，TCO の削減やセキュリティの強化を目指し，近年注目されている Diskless 構成をとる計算機システムの設計と実装について述べる.

Diskless 構成の特徴としては，

- 故障し易いハードディスクを利用せずに稼働するので，故障率が低い.
- サーバ側で OS などの更新をすれば良いため，更新時に端末が稼働している必要がない.
- クライアント側に HDD などのデータ記憶装置が無いので，盗難などにあったとしても重要な情報が流出しにくい.

など，TCO 削減やセキュリティの保持に寄与するとして期待が高い.

UNIX 系 OS では，ディスクレス構成は一般的に利用可能である [4, 5]. しかし，教育用計算機システムやクラスタシステムのようにほぼ同種類の複数の計算機を稼働させるようなシステムでは，通常ホスト毎に差がないディレクトリ (例えば/usr 以下) のみ共有し，それ以外はホスト毎に異なる領域を使用する形態になることが多い. その為，ホスト毎の差があるディレクトリであっても共通のファイルが大量に存在する場合 (例えば/etc 以下や Linux における/lib 以下等) には，クライアントごとに無駄なコピーを持つ負担が生じたり，ad hoc なシンボリックリンクを作成して共通化をはかったりする必要があり，稼働用のディスクイメージの更新の手間がかかる.

本稿では，端末として通常の Intel PC を利用した上で，Diskless 構成を unionfs を利用し，基本的に単一のディスクイメージを使用したシステムとして実装することを目標とする. また，実装には Linux と NetBSD の複数種類の OS を利用し，unionfs を用いた設計の実装可能性および実装の性能を評価する.

以降，2 節で本システムの設計について述べ，3 節で実装について述べる. また，4 節で実装したシステムについての評価について述べ，5 節で今後の課題に触れる.

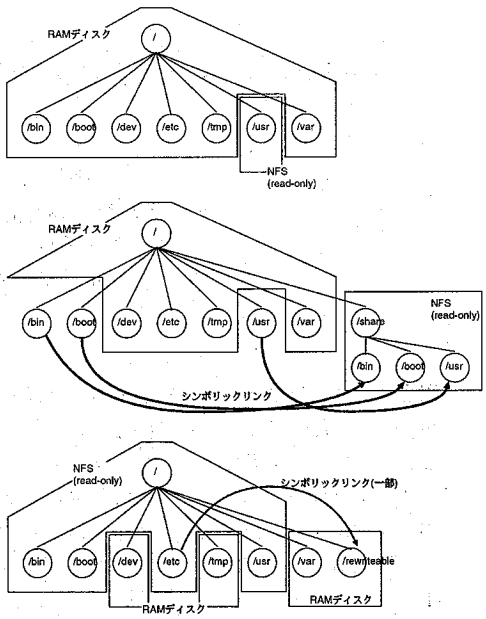

図 1: パーティション構成の例

2 設計

2.1 動作環境

まず，クライアントのハードウェアが，ローカルディスクの助けを借りることなく OS を起動する能力を持つ必要がある. 本稿では，端末は PXE(Pre Execution Enviroment)[6] 機能を持つパソコンを前提とする.

また PXE はサーバとして，特殊な応答を返す DHCP サーバと，ブートローダや OS のカーネルイメージを提供する tftp サーバが必要である. DHCP サーバとしては，ISC の dhcpd[7] の実装を利用すれば設定可能であり，tftp サーバは多くの UNIX 系 OS で標準装備されている.

2.2 パーティション構成

2.2.1 既存の構成法

Diskless 構成で稼働するパーティション構成としては,

1. root パーティションを RAM disk 上に配置 (RAMroot 方式)
2. root パーティションを NFS 上に配置 (NFSroot 方式)

という二種類が考えられる (図 1).

RAMroot 方式の場合, すべてのディスクイメージを RAM 上に展開して動作させることは稀であり, 通常, 共通に利用できるディスクイメージ (サブディレクトリ) を NFS マウントして利用する. また, NFSroot 方式の場合も, 運用中はディスクイメージの全ての部分がそれぞれのホストから書換えることができる必要があるわけではなく, 通常, 書換えられる部分と共通に読み込みのみで良いディスクイメージ部分とが存在するため, 分けてマウントすることが多い.

この共通に利用できる, すなわち読み込みのみのアクセスだけですむディスクイメージとして, 一般的な UNIX 系 OS では, /usr という単一のサブディレクトリ以下が考えられる. しかし, 最近の Linux では, それ以外にも, /boot, /lib, /etc が比較的大きく, 例えば Vine Linux 3.2 ではそれらをあわせて 100MB を越え, デバイスドライバの対応種類数の増加などに従って肥大化の傾向にある[1]. 従って, /usr のみ共通化するだけでは, 共通化のメリットが不足する場面がありえる.

この問題に対応する構成として,

1. 共通に利用できるディスクイメージ (サブディレクトリ群) を単一のマウントポイント以下にまとめ, そのディレクトリ以下へのシンボリックリンクを作成する (root-symlink 方式)
2. 共通に利用できるサブディレクトリ毎に, 個別の (サブ) ディスクイメージをマウントする. (root-eachmount 方式)

という二種類の方法が考えられる (図 1).

root-symlink 方式は, 一般によく使われる方式であるが, 共通のサブディレクトリ以下にアクセスするたびにシンボリックリンクを辿る為に, readlink(2) が頻繁に呼び出され負荷が高くなる傾向にある. また, root-eachmount 方式は, root-symlink 方式のようなシンボリックリンクを辿る必要は無くなるが, ディスクイメージサーバに対して, マウントポイントの数だけ NFS マウント要求が必要となる. この方式の場合にマウント要求が集中するタイミングは起動時だけではあるが, 教育用計算機システムやクラスタのように同時に多くのクライアントが起動するような環境では, NFS サーバに過剰な負荷がかかる可能性が高い[2].

2.2.2 共通化可能部分の分離

UNIX 系 OS では, /etc 以下にクライアント毎の設定やシステムに対応する設定が配置され, /var 以下にクライアント毎の状態やログといった情報が配置される. この 2 つのディレクトリツリーは, クライアント毎に別々に準備する必要があるとされる.

しかし, /etc 以下にはそのクライアントのホスト名やネットワークインターフェイスの IP アドレス, /etc/mtab のような個別に異なる設定だけではなく, 例えば /etc/resolv.conf や /etc/nsswitch.conf といったシステム内では共通になる設定ファイルがあったり, また, クライアント毎の設定でも, 例えば Linux での /etc/modules.conf といった同一ハードウェア構成なら同一となる設定も存在する.

逆に/var 以下にはそのクライアントのログやスプールといった個別に異なる情報だけではなく, 例えば Linux での/var/lib/rpm/以下にある導入されているアプリケーション情報といった, 比較的サイズが大きく, かつ, システム内で同一と考えられる情報が含まれている.

そのため, 単純に/etc, /var 以下をクライアント毎に準備すると, 無駄にコピーが発生したり, 同一に保つ為に全クライアント分の更新を実施する必要が生じてしまう. このような部分が多くなると, 更新にかかる時間が多くなってしまうため, 教育用計算機システムのように, 多くの台数を管理する必要がある場合に手間がかかってしまう.

また/dev 以下は, デバイスファイル自体は共通

[1] NetBSD 3.0_BETA では/usr 以外を全て集めても 25MB ほどである.

[2] nfsd は多くのクライアントからアクセス要求に耐えるように考慮されている場合が多いが, mountd はそのようなアクセスの集中はあまり考慮していないことが筆者らの経験上見受けられる

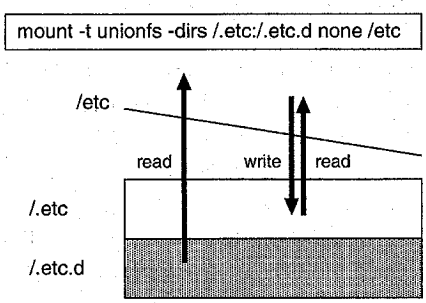

図 2: unionfs の例

利用可能ではあるが，ログインしたユーザにあわせて owner/group/permission を変更する必要のあるデバイス (tty やオーディオデバイス等) に対してはクライアント毎に個別に変更できる必要があるため，共通化はできない．/tmp 以下もクライアント毎に個別のファイルが生成されたりするが，起動毎に内容の継承をすることは不要であることが一般的であり，起動時に単に RAM ディスクを生成してマウントするだけで十分である．

つまり，単一イメージにできない部分は，/etc の一部，/var の大部分と，/dev, /tmp といえる．

ここで，/etc や/var の共通化について検討する．root-symlink 方式の場合，/etc であればサブディレクトリ自体は共通化して，変更が必要なファイルのみ再度 RAM ディスク上へのシンボリックリンクにするなどの方策が必要であり，readlink(2) の負荷がさらに増加することが懸念される．また，Linux の /etc/mtab のように，シンボリックリンクであるかどうかでカーネルの動作が変わるような実装もあり，注意深く動作確認の必要がある [2]．root-eachmount 方式の場合，/etc 以下は RAM ディスク上に作らざるを得なくなるが，/etc 以下に必要となるファイル群を RAM ディスク上に生成してからマウントするという方法では，kernel 起動直後から/etc/mtab のようなファイルは頻繁に更新される為，タイミングが合わないことになってしまう．

2.2.3 unionfs を用いた構成法

以上のような問題点を解決するために，本稿ではディスクレス構成の実装に unionfs を用いることを考えた．

unionfs[8] は，スタッカブルファイルシステムの一つであり，複数のファイルシステム (ディレクトリ) を重ねあわせることができる[3]．unionfs を用いれば，例えば読み込みのみのファイルシステム (1) の上に読み書き可能なファイルシステム (例えば RAM ディスク)(2) を重ねあわせることで，

- (1) の上にあるファイルはそのまま読み込み可能である．
- (1) の上にあるファイルを書換えようとすると，(2) の上にコピーされた上で処理され，見掛け上ファイルが書換え可能となる[4]．

という効果が得られる (図 2)．従って，例えば /etc の大部分を共通化したディスクイメージ (書き込み不可)) 上で提供しつつ，クライアント毎に書換えが必要な部分は unionfs の機能により書き換えてやれば，スマートに解決できる．

3 実装

2 節の設計に従い，NetBSD 3.0_BETA[11] および Vine Linux 3.2[10] 上で実装した．

3.1 NetBSD の場合

NetBSD の unionfs の実装は，

```
mount_union /dir1 /dir
mount_union -b /dir1 /dir
```

という形で利用すると，dir の上に (-b オプションを付与した場合は下に)dir1 を配置したものをマウントすることができる．従って，dir1 に RAM ディスクを指定し書換え可能にしたいディレクトリにマウントするだけで良い．

```
mount -t mfs -o -s=1024 none /.etc
mount_union /.etc /etc
mount -t mfs -o -s=1024 none /.var
mount_union /.var /var
```

ただし，NetBSD の unionfs の実装上，ディレクトリのパーミッション情報が，unionfs 構成での上になるディレクトリ側に支配される．そこで，起動時に，NetBSD がセキュリティ監査用に使用しているパーミッション情報確認ツール (mtree(8))

[3] NetBSD では標準で利用可能である．
[4] ファイルの削除もディレクトリファイルの書換えとみなすことができるので可能である．

を使って，ディレクトリパーミッション情報が正しいものになるように変更しておくように工夫した．

/dev 以下については，カーネル自身が自動的にRAM ディスクを生成し，必要なデバイスファイルをあらかじめ作成する機能が使えるようになっているので，それを使えば良い．

3.2 Linux の場合

3.2.1 必要最小限の unionfs の使用

NetBSD の場合と異なり，Linux で利用可能な [8] の実装は，

```
mount -t unionfs -o dirs=/d1:/d2 none /d
```

という形で利用し，/d2 の上に/d1 を配置したものを/d にマウントすることができる．従って，/d1 に RAM ディスクを指定し/d2 に読み込みのみの NFS パーティションを指定すればよい．/d2 と/d を同じに指定することは出来ないので，本来/etc にあるべきものを/.etc.d に配置した上で，以下のようにする．

```
mount -t tmpfs none /.etc
mount -t unionfs -o dirs=/.etc:/.etc.d
                                    none /etc
```

さらに上記の手続きを実行するまでに /etc の下が空では困るため，あらかじめ以下のような処理をしておく必要がある．

```
mv /etc /.etc.d
cd /etc; ln -sf ../.etc.d/* .
```

これにより，unionfs のマウントが実施される前でも，/etc 以下のファイルに読込みのみのアクセスが可能となる．

/dev 以下については，Linux では devfs と呼ばれるオンデマンドデバイスファイル生成システムを利用することも考えられるが，既存のアプリケーションとの連携状況があまり良くないことが知られている．従って，NetBSD のように小さな RAM ディスクをマウントし，必要なデバイスファイルをあらかじめ生成したものを利用することで対応する．

3.2.2 すべてに対する unionfs の使用

一般的な unionfs の実装では，あるサブディレクトリをその親ディレクトリに unionfs することはできないはずである．つまり，/ に対してあるディレクトリ以下 (/のサブディレクトリ) にマウントした RAM ディスクを unionfs を用いて上に配置することはできないため，前節のように必要なサブディレクトリ毎に unionfs を使用した実装を行っていた [3]．しかし，Linux に特有の initrd 機構 [9] を使用し，initrd 内で実行される初期実行スクリプト (linuxrc) 内で本来のルートファイルシステムのマウントが実施されるより前に，unionfs を使った準備を終わらせておけば (何故か) 期待された動作をする．

```
mount -t tmpfs none /unionfs
mount -t unionfs -o dirs=/unionfs:/sysroot
                                    none /sysroot2
pivot_root /sysroot2 /initrd/sysroot2
```

この方法では，どのサブディレクトリに書き込みが発生したとしても，unionfs の上側にある RAM ディスクへの書き込みとして動作するため，ディストロやアプリケーションが想定外のディレクトリ配下への書き込みを前提にしていたとしても問題が生じにくい．しかし逆に，書き込みが行われないことが想定されている領域に書き込みがあった場合に，一時的にせよ書き込み (書換え) が行えてしまうので，安全性の面を十分考慮する必要がある．

表 1: 評価環境

サーバ	SunBlade150 UltraSPARC IIe 650MHz, 2GB mem, 40GB(EIDE), 1000baseT(RTL6139S)
クライアント	IBM ThinkPad X40 PentiumM 1GHz, 768MB, 1000baseT(i82541GI)
ネットワーク	PCi PFX-08TXS

4 評価

4.1 起動時間

本稿で実装したシステムの性能を比較するために，OS の起動にかかる時間を測定した．まずは，クライアント単体で測定した．Linux で，unionfs を用いない状態と，ルートファイルシステム自体を unionfs で RAM ディスクを上にかぶせた状態とで比較した．OS 起動時間として，ブートローダ

自体の読み込みが完了した状態で，カーネルなどの読み込み開始を指示し始めてから測定を開始し，login: が表示されるまでをストップウォッチで計測した．

このとき，unionfs が無い場合が約 65 秒に対して，unionfs ありでは約 58 秒で起動した．これは，unionfs のオーバヘッドよりも，新規ファイルの生成が RAM ディスク上になることによるスピードアップが大きかったことによるものと考えられる．

4.2 ファイル数

NetBSD の場合，起動直後で，/.etc と/.var をあわせても i-node 数で約 150，ファイル容量として約 200kByte の使用量で済んでいる．また，Linux の場合，/.etc と/.var で i-node 数で約 370，ファイル容量として約 2.5MByte の使用量であった．

5 まとめ

本稿では，大学などの教育用計算機システムに適し，TCO 削減やセキュリティ強化に寄与するとされている Diskless 構成をとるコンピュータシステムを，unionfs を使用することで設計を行い，Linux と NetBSD 上で実装しその性能を評価した．本稿ではディスクレスシステムとして実装をおこなったが，ディスクイメージの基本部分を書き込み不可として稼働させているという観点からは，以下のような応用も考えられる．

CD/DVD-ROM 媒体から直接動作するシステム
自分の好みの UNIX 系 OS を Knoppix と似たような使い方をすることが可能になる．

ローカル HDD を読込みのみで利用するシステム
書き込みが無いため，電源を自由に落とすことができ，さらにその後は元に戻るため，堅牢性が高く KIOSK 端末などに有効である．

Xen などの VM 上での多数のゲスト稼働
ゲストマシンの数だけのゲスト OS イメージを準備する必要がなくなる．

コンピュータ管理の教育
初期インストール済み状態からそれぞれの設定演習が可能になる．

今後の課題として，運用を見据えた場合の性能比較や他の OS での同様の仕組の調査検討などが挙げられる．

謝辞

有益な意見や協力を戴いた，有限会社ヴァインカーヴの鈴木大輔さん，やまだあきらさん，松林弘司さん，に感謝します．

本研究の一部は，科学研究費補助金 基盤研究 (C)(2) 課題番号 17500050 による．

参考文献

[1] 桝田他：“大規模分散ネットワーク環境における教育用計算機システム”, 情処学会会誌, pp.225–281, Vol.45, No.3, 2004.

[2] 桝田, 齊藤：“ディスクレス環境の教育用計算機システムに適した Linux システムの実装”, 情処学会 DSM シンポジウム 2004, pp., 2004.

[3] H.Masuda, A.Saitoh, S.Yasutome and M.Nakanishi: “Diskless Linux system with unionfs for an educational computer center”, SIGUCCS'05, November 6-9, 2005.

[4] Diskless Linux Mini Howto, http://www.linux.or.jp/JF/JFdocs/archive/Diskless-HOWTO.html.

[5] Diskless NetBSD HOW-TO, http://www.jp.netbsd.org/Documentation/network/netboot/.

[6] Intel 社, Preboot Execution Environment (PXE) Version 2.1, ftp://download.intel.com/labs/manage/wfm/download/pxespec.pdf/.

[7] ISC, DHCPD, http://www.isc.org/sw/dhcpd/.

[8] A Stackable Unification File System, http://www.fsl.cs.sunysb.edu/project-unionfs.html.

[9] 起動用 RAM disk(initrd), http://www.linux.or.jp/JF/JFdocs/kernel-docs-2.2/initrd.txt.html.

[10] Vine Linux, http://www.vinelinux.org/.

[11] NetBSD, http://www.netbsd.org/.