2006－ＤＳＭ－41（15）
2006／5／12

Xenとunionfsを用いたサーバファーム運用の可能性の検討

桝田 秀夫† 齊藤 明紀††

† 京都工芸繊維大学 情報科学センター
〒606-8585 京都市左京区松ヶ崎御所海道町
†† 鳥取環境大学 情報システム学科
E-mail: †h-masuda@kit.ac.jp, ††saitoh@kankyo-u.ac.jp

あらまし TCO削減の為に，単一の計算機上で複数のOSインスタンスを稼働させる技術が注目されてきており，中でもオープンソースのXenが脚光を浴びてきている．ネットワークサービスを提供する場合に，複数のOSインスタンスを利用することで冗長度を高めたり，同一OSインスタンス上のサービスアプリケーション間の影響を減らす為に，サービス毎に別のOSインスタンスを使用する構築方法をとることがしばしば行われている．これらの手法では，OSインスタンスごとに，アプリケーションの追加，削除，パッチ適用を行っていては，TCO軽減効果が薄くなってしまう．そこで，本報告では，Xen上で単一のOSイメージをunionfsを用いて個々のOSインスタンスが共有する方式による保守作業の一括化を提案する．また，unionfの適用形態によるTCO削減とパフォーマンスの差異についても検討を行う．

キーワード サーバファーム，仮想計算機 (Virtual Machine), Xen, unionfs, TCO削減

Some implementation of a server farm by Xen and Unionfs

Hideo MASUDA† and Akinori SAITOH††

† Center for Information Science, Kyoto Institute of Technology.
1-32 Machikaneyama Toyonaka OSAKA, 560-0043, JAPAN.
†† Department of Information System, Tottori University of Environmental Studies.
E-mail: †h-masuda@kit.ac.jp, ††saitoh@kankyo-u.ac.jp

Abstract To reduce the total cost of ownership, the virtual machine (VM) technology is widely discussed. "Xen" is one of the open-source VM technology. For the network services, we often use the clustering technique with same OS instances and/or the separating the OS instances between each services getting rid of the service applications interference. In these techniques, it is very troublesome if administrators add/remove/patch the applications for each OS instances. In this paper, we propose a server farm configuration method with Xen and unionfs technologies. In our approach, one VM uses a small unionfs file system and underlying single shared OS image. We also show the evaluation of the performance and how reduce the TCO.

Key words Server farm, Virtual Machine, Xen, unionfs, TCO

1. はじめに

近年，計算機ハードウェアの高性能化に伴い，単一の計算機上で複数のOSインスタンスを稼働させる仮想計算機技術が再び注目されてきている[3]～[8]．また，サーバ計算機によるネットワークサービスの提供をできるだけ停止させない為に，ハードウェアを多重化してクラスタ化を行ったり，サービス毎に別のOSインスタンスに分割するようなシステム構築も良く行われる．例えば，メールサーバとWWWサーバ，DNSサーバをそれぞれ別の計算機システム上で稼働させておき，さらにサービス毎に同じ機能を持つ計算機システムを複数準備してクラスタ化しておく構築方法が考えられる．この方法では，メンテナンスの際に，クラスタの一部の計算機毎に停止等を実施することでネットワークサービスをほとんど停止させないようにすることができる．またパッチ等のサービスへの影響評価も，同一OSインスタンス上のサービスアプリケーション間の依存関係を考慮する必要が無くなり，評価作業が容易になる．

このように，多くのOSインスタンスを稼働させる為に，多数の計算機を用いることとすると，ハードウェアのメンテナンスコストや電気代等，運用管理コストの増大につながってしま

う．ここに，仮想計算機技術を用いると，ハードウェアコストなどを削減することが期待される．それでもなお，OS インスタンスごとに，アプリケーションの追加，削除，パッチ適用を行っていては，TCO 軽減効果が薄くなってしまう．

そこで，本報告では，仮想計算機技術として Xen を用いた上で，単一の OS イメージを unionfs を用いて個々の OS インスタンスが共有する方式による保守作業の一括化を提案する．

2. 仮想計算機技術

本節では，既存の仮想計算機技術について得失を述べる

2.1 計算機自体を仮想化する方式

汎用の OS 上で計算機ハードウェア自体をエミュレートし，通常の OS を稼働させようとするアプローチであり，Full Virtualization と呼ばれる．このアプローチを取るものとしては，VMware [3], Virtual PC [4], QEMU [5] などがある．これらは PC/AT 互換機をエミュレートする実装(注1)をとっている為，仮想計算機上で稼働させる OS(カーネル) にはほとんど手を入れる必要がない．しかしながら，ネットワークやディスクなどのデバイスに対するアクセス時にはどうしても大きなオーバヘッドがかかってしまう．

2.2 OS カーネルをアプリケーション化する方式

ユーザプロセスとして，OS のカーネルを実行させるようにするアプローチである．このアプローチを取るものとしては，coLinux [7], User-Mode-Linux [6] などがあり，Microsoft Windows や Linux 上のーアプリケーションとして OS インスタンスを，ベースの OS とは独立して稼働させることができる．計算機ハードウェア自体をエミュレートするアプローチよりはオーバヘッドが少ないが，OS(カーネル) に手を入れる必要があるため，稼働させる OS の種類が限定される上，ネットワークやディスクなどのデバイスへのアクセスがユーザプロセス空間で実行されるため，パフォーマンスが犠牲になる部分が存在する．

2.3 仮想計算機に特化した方式

仮想計算機を実現する専用のモニタを持ち，OS のカーネルにもこのモニタに対応した形での実装を行うアプローチであり，Paravirtualization と呼ばれる．このアプローチは，Xen [8] などがその例である．仮想計算機を実現することに特化しているため，仮想計算機環境でのパフォーマンスに優れている．Xen は，Linux や NetBSD に対応している上，オープンソースであり，さらにバージョン 3 では，Intel 社の Vanderpool Technology [9] に対応した Microsoft Windows の稼働も可能であると言われている．

本稿では，仮想計算機の方式として Xen を用いることとする．

3. 設　　計

多数の似通った OS インスタンスの運用管理の負荷を軽減するために，文献 [1], [2] で発表した unionfs を用いたアプローチを採用して設計を行う．

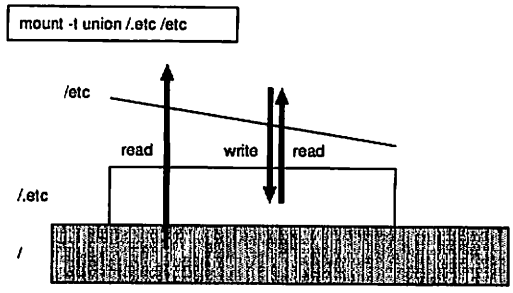

図 1　unionfs の概要

3.1 OS イメージの単一化

クラスタ化されたサーバ間だけではなく，メールサーバや Web サーバ，DNS サーバとサービス機能が異なったサーバの間であっても，基本 OS イメージの多くの部分は共通であることがほとんどである．さらに，サービス機能毎に異なるアプリケーションの導入が必須ではあるが，ほとんどの場合，アプリケーションの導入に加えて，設定ファイルを適切に設定して初めてサーバとして機能している．

従って，基本 OS イメージに加えて，それぞれのサービス機能に必要なアプリケーションを導入した状態を新たに**拡張基本 OS イメージ**と考え，これを共有することとする．

3.2 ホスト固有部分の構成

文献 [1] にあるように，ホスト固有のファイル群は，ホスト固有のファイルシステムを構成した上で，unionfs を用いてファイルシステムを重ね合わせれば，拡張基本 OS イメージは読み込みのみの状態で問題なく運用が可能となる．図 1 は，/.etc を/etc ディレクトリの上に重ねて unionfs でマウントした場合の例である．/etc は root ファイルシステムとしてマウントされた拡張基本 OS イメージに含まれている．/etc 配下のファイルにアクセスした場合，実際には/.etc にあるファイルがアクセスされる．/.etc に存在しないファイルを読み出した場合に限り，もともとの/etc のファイルがアクセスされる．

基本拡張 OS イメージでは，OS インスタンスごとに変える必要のある設定ファイルや書き換えが必要なファイルはすべて/etc 配下か/var 配下に置かれるようにする．このようにすることで，不必要なディレクトリへの書き込みを物理的に禁止することが可能となる．そのため，アプリケーションやデーモンプログラムファイルへの不正な書き込みアクセスが行われるような事態が発生しても，実際には書き換わることはなく，サービスの安全性が高まるとも考えられる．

さらに，設定ファイル部分のコピーを作成したものを用意し，それをパッチを適用した拡張基本 OS イメージに重ねあわせることで，パッチ適用後の動作確認も比較的容易に実施できると考えられる．

(注1)：PowerPC による Macintosh をエミュレートするものなどもあるが，本稿では，稼働 OS の種類が多く比較的コストパフォーマンスのよい PC/AT 互換機のみを考慮する．

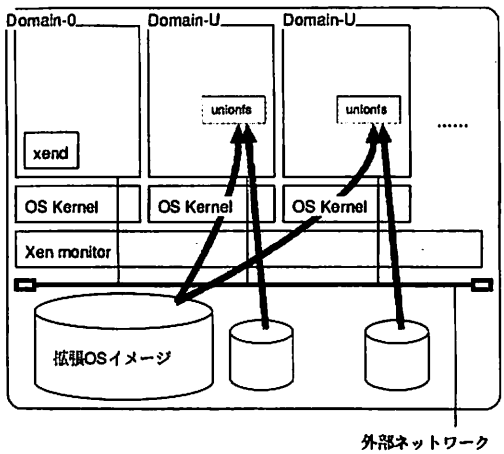

図 2 Xen を用いたサーバファームの構成図

```
1  /dev/xbd0a / ffs ro 0 0
2  /dev/xbd1a /.etc ffs rw,softdep 1 1
3  /dev/xbd1b swap swap sw
4  /dev/xbd1e /tmp ffs rw,softdep 1 2
5  /dev/xbd1f /.var ffs rw,softdep 1 2
6  /.etc /etc union rw,hidden
7  /.var /var union rw,hidden
```

図 3 /etc/fstab の概要

/etc/xen/domainU1

```
1  kernel = "/netbsd.XENU"
2  memory = 512
3  name = "domainU1"
4  vif = [ 'mac=aa:00:00:00:00:01, bridge=bridge0' ]
5  disk = [ 'file:/xen/domainU1,xbd1,xbd1d,rw',
       'phy:/dev/vnd0d,xbd0d,r' ]
6  root = "/dev/xbd0a"
```

図 4 Domain-U の設定概要

4. 実　　装

NetBSD 3.0 と Xen 2.0.7 (注2)で実装した.

4.1 Xen の適用方法

Xen では，仮想計算機モニタである xen monitor の上で各仮想 OS インスタンスが動作するが，最初に起動する OS インスタンスが "Domain-0" と呼ばれ，特別な機能を持つ．それ以外の仮想 OS インスタンスは "Domain-U" と呼ばれる．Domain-0 では，Domain-U を新たに作成したり制御を行うことのできる xend というデーモンが稼働している．そこで，本実装では，ネットワークサービスに使用する仮想 OS インスタンスは Domain-U 上であるとし，Domain-0 は単一 OS イメージのメンテナンスをはじめとする各種制御専用として利用することとした (図 2).

4.2 共通部分とインスタンス固有部分の分離

前節で述べた通り，unionfs を用いてインスタンス固有部分を共通部分に重ね合わせる方式を採用する．インスタンス固有である必要のあるディレクトリを詳細に調べたところ，以下のディレクトリが該当した.

/etc ：このディレクトリには，各種設定ファイルが保存されているので，unionfs を用いてインスタンス独自のディレクトリを重ね合わせることとする．文献 [1] では，メモリファイルシステムを重ね合わせ，起動時に必要なファイルを生成する方式を取っていたが，本稿ではサーバであることから，生成するスクリプトを作成し更新する方法は管理の手間が大きく，他のサーバへの予期しない影響が混入する恐れがある．従って，通常のディスクファイルシステムを重ね合わせることとし，ホスト毎のファイルをあらかじめ準備しておくことや，リブートしても消えないように考慮した．逆に，単一の Xen システム上で共通にしてよい設定ファイルは，拡張 OS イメージ側に適用しておけばよい.

/var ：このディレクトリには，ログファイルやスプールが保存されている．当然ながらリブートで消えるようなことがあってはならず，また，サーバであるためログファイルやスプールは大きな容量が必要となると考えられる．文献 [1] でも触れているが，完全にホスト毎に異なる状態であるべきものだけではなく，パッケージ管理システムのデータベースも/var/db/pkg 以下などに存在しており，これらは共有化するべきである．従って，このディレクトリも，/etc と同様に通常のディスクファイルシステムを重ね合わせるようにする.

/dev ：文献 [1] でも触れた通り，NetBSD にはカーネル読み込みが終了して/sbin/init を起動したときに，/dev/console が無ければ自動的に必要最小限のメモリファイルシステムを作成してデバイスファイルを作成する機構が備わっている．従って，実装上特に工夫は必要ない.

/tmp ：/tmp 以下は通常ブート時に掃除をする設定が標準的であり，リブート後も残したい一時ファイルは/var/tmp 以下に配置することが多い．しかしながら，サーバの場合はある種の証拠となるファイルが/tmp 以下に残っている可能性や，サービスアプリケーションにメインメモリをできるだけ与えたいという要望が多いと考えられる．そこで，/tmp も通常のディスクファイルシステム上に配置する．ここでは，共有化するべきファイルはないので，unionfs を用いる必要はない.

swap ：多数の仮想 OS インスタンスを稼働させたいとなると，潤沢なメインメモリを各 Domain-U に割り当てることは物理的な制約もあり難しい．従って，通常のディスクファイルシステムとして swap 空間も割り当てる構成とするのが望ましいと考えられる.

4.3 NetBSD での実装の概要

(1) まず，Domain-0 は制御用として利用するとしているので，拡張 OS イメージとは無関係に NetBSD の標準セットをインストールし，xend などの Xen の制御用ツールのみパッ

(注2)：執筆時点での最新版は 3.0.1 であるが，実験の都合上 2.x 系列の最新版である 2.0.7 を使用している.

表1 評価環境

サーバ	Athlon64 マシン Athlon64 3500+, 2.2GHz, 2GB mem, 250GB x2(EIDE, software RAID1), 1000baseT(INTEL PRO/1000 MT)

ケージシステムからインストールする.

(2) その上で, 拡張 OS イメージ用に, 1 つのパーティションを準備して, NetBSD の標準セットと, ネットワークサーバに必要なアプリケーションをパッケージシステムなどを利用してインストールする. このパーティションは, Domain-U から見ると 1 台目の読み込み専用のディスク装置に見えるように設定する.

(3) 次に, Domain-U 用に, 必要な数だけ Domain-0 上で 1 つのパーティション (個別イメージ) を準備する. このパーティションは, Domain-U から見ると 2 台目のディスク装置に見えるように設定する.

(4) 個別イメージを, a (/etc 用), b (swap 用), e (/tmp 用), f (/var 用) という形で複数のパーティションに分割する.

(5) 拡張 OS イメージ上の/etc/fstab を, 図 3 のような形で記述する.

このような準備をした上で, 図 4 のような設定で Domain-U を稼働させる.

5. 評　　価

前節の方針で実装したサーバファームシステムについての評価を行う. 評価に用いた計算機環境を表 1 に示す.

5.1 比較項目

本稿での実装の大きな特徴である, unionfs の効果と拡張基本 OS イメージの共有化の効果について, それぞれ比較を行う.

5.1.1 unionfs の効果

unionfs の効果を調べるために, まず, 拡張基本 OS イメージの共有を unionfs によって行う場合 (unionfs) と unionfs を使わずに拡張基本 OS イメージを Domain-U の数だけコピーを準備する場合 (non-unionfs) とで比較をする.

unionfs を使う場合は, unionfs 経由でのファイルアクセスの分だけオーバヘッドがあると考えられるが, 大部分を読み込みのみモードでマウントしているためアクセスタイム属性の更新がなく実質的に高速になる可能性があると考えられる. また, 同時起動であれば, 複数の Domain-U から OS イメージ中の似たファイルへのアクセスが比較的時間のずれなく発生すると考えられるため, Domain-0 側でのファイルキャッシュ効果などが有効に働くとも考えられる.

5.1.2 共有アクセス方法の効果

次に, 拡張基本 OS イメージへの共有アクセス方法について, Diskless システムのように NFS root として動作させる方法 (nfs) と ufs2 のファイルシステムをそのまま読み込みのみモードでアクセスする方法 (ufs) とで比較をする.

nfs を使う場合は, 元々複数のホストからの読み書きが想定されているため, 拡張基本 OS イメージの更新を一つの Domain-U から実施してもファイルシステムとしての不整合は生じないが, その分低速である. ufs を直接使う場合は, 通常ファイルシステムを複数の OS から同時に読み書きされることは想定していないため, 複数の Domain-U 間で共有している際に, 別の Domain-U からそのファイルシステムに書き込みが発生すると不整合が生じる可能性がある. しかし, 拡張基本 OS イメージの更新の際には, まずその複製を作成した上で, 専用の Domain-U を立ち上げて更新を実施し, 更新完了後にそれぞれの Domain-U を一つずつ順番に, shutdown しては更新後の拡張基本 OS イメージで boot するという形で reboot すれば, その問題は発生しないと考えられる.

表2 起動時間

同時起動 Domain-U 数	1	2	3	4	
unionfs + ufs	9.0	10.0	14.7	17.5	(秒)
unionfs + nfs	13.0	14.5	17.3	18.7	(秒)
non-unionfs + ufs	7.0	9.7	12.2	14.2	(秒)
non-unionfs + nfs	12.0	15.0	17.0	18.0	(秒)

5.2 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. OS 起動時間として, Domain-U を (同時に) 起動させた時刻を記録した後, 各 Domain-U のコンソール上で login:プロンプトが表示される直前に実行されている date コマンドの出力表示をを確認し, その平均値を取る. OS の起動時に稼働させるデーモンは, rpcbind, amd, statd, lockd, powerd, routed rwhod, sendmail, inetd, cron の 10 種類とした. 結果を表 2 に示す.

これらの結果から, unionfs 方式の方がオーバヘッドが少し大きいと予想される. また, ufs 方式と nfs 方式ではやはり ufs 方式の方が高速にアクセスできると予想される.

5.3 運用の手間

次に, 本稿で実装したシステムの運用の手間についての評価を行う. 運用の手間の違いがある部分を考えると, 以下の点が差異となる.

- unionfs を使った OS のセットアップにかかる作業

unionfs を用いた場合は, 拡張基本 OS イメージは一つ準備するだけで良く, 仮想 OS インスタンスを追加する場合には, ホスト毎の差分パーティションを追加する台数分準備して設定をするだけで済む. unionfs を用いない場合は, それらをすべて 1 から実施することになるが, 既存の仮想 OS インスタンスからまるごとコピーした上でホスト毎に対応する設定を変更すればよいので, 設定変更部分としての手間は大きくは変わらない.

しかし, unionfs を使う場合には, /etc の配下への設定変更が, それぞれのホスト毎にするべきものなのか, それとも拡張基本 OS イメージ側に適用するべきなのかを検討する手間が増える.

- パッチの適用にかかる作業

unionfs を用いた場合は, 一つの拡張基本 OS イメージにパッチを適用するだけでよいが, そうでない場合は, すべての OS イメージにパッチを適用する必要がある. また, パッチの適用

によってネットワークサービスに不具合がでないかを確認したい場合，unionfs を用いた場合は，拡張基本 OS イメージのコピーに対してパッチを適用したうえで，確認したいサービスの稼働しているホスト毎のファイルシステムのコピーを unionfs で重ねあわせることで，非常にスマートに動作確認が可能となる．しかしながら，unionfs を用いない場合だと，少なくともサービスの数だけ，場合によっては各ホスト毎に動作確認をする必要があるため，かなり負荷が高い．unionfs を用いる場合でも，パッチの適用で /etc 以下の設定ファイルの一部が書き換わった場合，あるホストでは別途書き換えているような状況には簡単には対応できないので，その整合性を取る手間が増える．

6. 考　　察

構築したシステムについて，いくつかの問題が判明しており，今後の調査研究，検討が必要である．

- NetBSD の unionfs の実装に起因する問題として，読み込みのみモードでマウントしたファイルシステム (下側) 上に読み書き可能なファイルシステム (上側) を unionfs で重ねた場合，下側にのみ存在するファイルへの書き込みに対して「読み込みのみ可能なファイルシステムである (EROFS)」というエラーが発生する．当該ファイルを rm(1) を使用して強制的に削除することは可能であり，一旦ファイルエントリを削除した上でファイルを作成すれば，上側のファイルシステム上でファイルが作成されているため，通常の操作が可能となる．この問題が障害になるのは/etc 配下のいくつかのファイルであるが，あらかじめ対象ファイルを拡張基本 OS イメージ側から削除しておくか，起動時に "mv -f /etc/FOO /etc/FOO- && cp -p /etc/FOO- /etc/FOO && rm -f /etc/FOO-" という一連の操作を実施しておくことでも回避できる．しかしながら，これを多用すると /etc 以下のファイルがホスト毎に共通であるかどうかに関わらずコピーを持つ方式となり，unionfs の利点が活かされない．
- 拡張基本 OS イメージ自体の更新を，ある Domain-U もしくは Domain-0 から実施する場合，サービスを提供している他の Domain-U からは読み込みのみモードでマウントしているため，参照しているファイルが消えたり更新されたりする場合がある．特にファイルのキャッシュの方式によっては問題が出る可能性がある．
- 拡張基本 OS イメージ上の/etc 以下の更新が，ホスト毎で別途書き換えているファイルに対して起こった場合，それは上側のファイルシステム側にファイルができてしまっているため，その変更点が適用されない．
- 拡張基本 OS イメージを共有アクセスする場合，そのイメージの更新を稼働中に更新することを考えなければ，ufs2 のディスクイメージをそのまま共有する方式が高速である．しかし，その状態で拡張基本 OS イメージに対して変更を加えた場合，Domain-U 側から見てファイルシステムの不整合が発生する可能性がある．そのような状況を考慮すれば NFS root 方式が良いと考えられるが，どうしてもパフォーマンスがでにくい問題がある．

7. ま と め

本稿では，単一計算機上で複数の OS インスタンスを稼働させる仮想計算機技術の一つである Xen を用い，さらに複数のディレクトリを重ねあわせることのできる unionfs を組み合わせることにより，システムの運用管理の効率化を狙える実装を行い，その評価をした．

今後の課題として，実際の運用現場への適用や他の仮想計算機技術や他の稼働 OS における，同様の仕組みの調査検討などが挙げられる．

謝　　辞

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．

文　　献

[1] 桝田, 齋藤：“unionfs を用いたディスクレスシステムの実装とその評価”, 情処学会 DSM シンポジウム 2005, pp., Dec 01-02, 2005.
[2] H.Masuda, A.Saitoh, S.Yasutome and M.Nakanishi: “Diskless Linux system with unionfs for an educational computer center”, SIGUCCS'05, Nov 06-09, 2005.
[3] VMware, http://www.vmware.com/.
[4] Virtual PC, http://www.microsoft.com/japan/windows/virtualpc/.
[5] QEMU Emulator, http://fabrice.bellard.free.fr/qemu/.
[6] User-mode Linux Kernel, http://user-mode-linux.sourceforge.net/.
[7] Cooperative Linux, http://www.colinux.org/.
[8] Xen, http://xen.cl.cam.ac.uk/Research/SRG/netos/xen/.
[9] Intel Virtualization Technology, http://www.intel.com/technology/computing/vptech/.
[10] The NetBSD Project, http://www.netbsd.org/.