Virtual Machineを活用した大規模教育用計算機システムの構築技術と考察

丸山 伸[†] 最田健一[†] 小塚真啓[††]
石橋由子[†††] 池田心[†††]
森幹彦[†††] 喜多一[†††]

本論文では，各端末から Windows と UNIX を同時に利用できる教育用計算機システムにおける，Virtual Machine を用いた新しいシステム構築技法について論ずる．まず，SMB プロトコルによるファイルサービスを，サーバ上で集約して行うのではなく利用者端末内で分散して行う手法を，端末数が 1,000 台を超す大規模なシステムにおいても両 OS から同時に利用できるホームディレクトリの構築手法として提案する．ついで，Virtual Machine 上で稼働する UNIX の起動時間の短縮と設定の初期化のためにハイバネーションを利用しつつ，セキュリティパッチの適用やシステム設定の変更を可能とする手法を提案する．これらの提案に基づく大規模教育用計算機システムを実際に構築し評価した．その結果，提案手法によりそれぞれ，(1) 実用上問題のない速度と安定性でホームディレクトリを提供できていること，(2) UNIX のファイルの変更手法を更新するべきファイルの分量や緊急度により選択することで，実用に耐える時間で，更新作業を含めた UNIX の起動が実現できることを確認した．

A Virtual Machine Solution for Large Scale Educational Computer Systems

Shin Maruyama,[†] Ken-ichi Saita,[†] Masahiro Kozuka,[††]
Yoshiko Ishibashi,[†††] Kokolo Ikeda,[†††] Mikihiko Mori[†††]
and Hajime Kita[†††]

This paper discusses a new method to construct educational computer systems with virtual machine technology, which serve two operating systems, i.e., Windows and UNIX on each terminal simultaneously. First, we propose a construction technique of the home directory, which can be simultaneously accessed from the both OSs in large-scale systems which consist of more than 1,000 terminals. In the proposed technique, SMB protocol needed for Windows is handled not by a single server but by each user terminal separately. We also propose a technique of performing application of security fixes and making a change of a system setup on that spot while we use hibernation in order to shorten the starting time of UNIX working on a virtual machine. A large-scale educational computer system based on these proposals was built, and its performance was evaluated. Consequently, we have confirmed that, (1) The home directory is offered with its performance satisfactory speed and stability in practical educational usage. (2) By choosing the technique of changing files of UNIX considering the quantity and urgency of update, the proposed technique can be used also by the large-scale system, and the starting time of UNIX including updating work is also within the range of practical usage.

1. はじめに

理工系と文科系とが共存する大学等において，授業や自習で学生が利用する教育用計算機システムを設計する際には，Windows と UNIX 系 OS（以下，UNIX とする☆）の双方を利用できるシステムであることが前提となる．Windows は現在最も広く普及している OS であり，学生が自宅等において最も慣れ親しんでいるだけではなく，卒業後に利用する可能性が最も高

† 京都大学大学院情報学研究科
Graduate School of Informatics, Kyoto University
†† 京都大学法学部
Faculty of Law, Kyoto University
††† 京都大学学術情報メディアセンター
Academic Center for Computing and Media Studies, Kyoto University

☆ ここでは，BSD 系システム，Linux，UNIX を総称して，UNIX という語を用いることにする．

いという点からも要望がある．一方 UNIX は以前から大学における情報処理教育において理工系を中心に広く利用されているというだけではなく，そのソースコードが公開されているという点からも，教育上の必要性が高い．

この要求を満たすための Windows と UNIX の両方を利用できるシステムを構築する手法は，以下のように大きく分類される：(1) Windows 端末，UNIX 端末の両方をそれぞれ物理的に用意する[1]，(2) 一方の OS は端末上のものを利用し，もう一方の OS はネットワーク越しに利用する[2]，(3) 電源を投入直後にどちらの OS を利用するのかを利用者が選択する（Dual Boot 方式）[3]．

(1) の手法は，OS ごとでの端末の需要が講義の内容や進度，学生の利用状況により変動するため端末に無駄が生じ，限られた端末設置スペースの有効活用ができない．

(2) の手法は，ネットワーク越しに利用される側の OS は，同時に複数の利用者にサービスすることを前提に性能を見積もる必要があり，サーバへの要求性能や導入コストが高くなるといった問題が生じる．

さらに (3) の手法では，手法 (1) や (2) にあるような問題は生じないが，多数の端末群の遠隔管理を可能にするためには，遠隔管理を行うための機構を各 OS に対応して用意する必要が生じ，管理上の手間が多い．また，連続する講義において異なる OS を利用する際に，授業の合間のわずかな時間に再起動する必要も生じる．さらには，Windows と UNIX の両方を同時に利用したいという要求に対応することができないという問題もある．

ところで，Virtual Machine はソフトウェアにより計算機をエミュレートする技術であり，古くは IBM360 等から使われてきた．近年クライアント端末として使われる IBM-PC 互換機が高性能になり，CPU やメモリに余裕が出てきたことから，IBM-PC 互換機上で Virtual Machine 技術により IBM-PC 互換機のエミュレートが実用的に利用可能となってきた．また，デバイスのエミュレーション等がパフォーマンス低下の原因ともなりうるが，実際には実用的な性能が出ることが示されている[4]．

このような流れを受けて，Windows と UNIX との両方を利用できるシステム構築の第 4 の手法として，Virtual Machine 技術を利用し Windows と UNIX とを同一の端末上で動作させる手法が注目を集め，構築事例も報告されている[5)~7)]．これらのいずれの事例でも示されているように，計算機システムを教育目的に利用する際には，Virtual Machine 技術は非常に有用である．

しかしながら，教育用計算機システムはその対象となる利用者である学生数が多いことがめずらしくなく，端末数が数百台以上となることもめずらしくない．さらには端末を 24 時間利用できることを要求されるため，いったん稼働を始めたシステムに対するセキュリティ問題の修正等は，可能なかぎり遠隔から自動的に行われることが期待される．

このような教育用計算機に特徴的な規模や制約において Virtual Machine を用いた教育用計算機システムの構築する際には，いくつかの問題を解決する必要がある．

まず，Windows と UNIX のそれぞれにホームディレクトリを提供するにはいくつかの手法が存在するが，これらを統合して共通のホームディレクトリを利用しようとするためには，従来のいずれの構築手法を用いた場合においても大規模な運用は困難である．

また，Virtual Machine を利用したシステムにおいては，一般に CPU やメモリ等のリソースを両 OS で共有するため処理が遅くなる．とりわけ起動やログインに要する時間が長いと，授業時間ごとに利用する学生が入れ換わる教育用計算機システムでは，運用上の大きな問題となる．

そこで，本論文では Virtual Machine を用いた大規模な教育用システムを構築するうえで生じる上記 2 つの問題に対して，新たな解決手法を提案する．

以下，2 章では Virtual Machine の特徴を整理し，Virtual Machine を用いた大規模な教育用計算機システムを構築する際に発生する問題を示す．3 章ではクライアント上でファイル共有プロトコルの変換を行う手法により，大規模なホームディレクトリサービスを提供する手法を提案する．さらに，4 章では端末の起動やログインに要する時間を短縮するために Virtual Machine のハイバネーションを利用する手法を採用した際に，新たに発生する「パッチの適用手順の問題」に対して，パッチのサイズにより手順を変更する手法を提案する．ついで，筆者らは京都大学学術情報メディアセンターの教育用計算機システムの更新にあたり，本提案手法を適用した．5 章ではこの事例を紹介し，6 章では構築したシステムに評価を与え，本論文の提案手法が実用的であることを示す．

2. Virtual Machine を用いた教育用計算機システム

2.1 Virtual Machine の特徴

近年，端末の CPU やメモリの高性能化は著しく，その性能を活かし，端末上で Virtual Machine 技術を利用して，ソフトウェアにより IBM-PC 互換機をエミュレートできるようになった．以下，Virtual Machine を動作させている OS を HostOS，Virtual Machine 上で動作する OS を GuestOS と呼ぶ．

Virtual Machine においては，すべてのデバイスはエミュレーションにより実装されている．そのため，Virtual Machine 内でのネットワークの扱いは一般には難しく，またエミュレーションによる速度低下が生じる可能性がある．

しかし近年の Virtual Machine 技術の進歩により，パフォーマンスを犠牲にすることなく柔軟な構成をとれるようになっている．たとえば商用の Virtual Machine の 1 つである VMware では以下の 3 つのモードが利用できる．

(1) ブリッジモード GuestOS に HostOS とは異なる IP アドレスを割り当て，HostOS のネットワークインタフェースを GuestOS からも利用することができる．このため，ネットワーク越しに直接 GuestOS にアクセスすることができる．

(2) NAT モード HostOS が GuestOS の属するネットワークのデフォルトルータとしての役割を行いつつ，外部へと送出されるパケットのアドレス変換（NAT）を行って，GuestOS から外部への接続を可能とする．外部から GuestOS に直接アクセスすることはできない．

(3) Virtual Network モード HostOS と GuestOS とだけが属するネットワークを仮想的に構築する．HostOS と GuestOS との間で送受信されるパケットは端末外に出ることはないため安全性が高い．

また，GuestOS はソフトウェアによりエミュレートされる PC 上で動作するため，Virtual Machine ソフトウェアに対して指令を送ることで，Virtual Machine 上で稼働する GuestOS の動作をいったん停止したうえで，動作中の状況の保存と動作の再開ができる（ハイバネーションと呼ぶ）．

さらには，動作中の状況を保存しておき，後から何度もその状態から動作を再開することもできる（Repeatable Resume と呼ぶ）．

2.2 Virtual Machine 技術を利用した教育用計算機システムの問題点

Virtual Machine 技術を用いた Windows と UNIX とが共存する教育用計算機システムの構成法には，(1) HostOS を Windows，GuestOS を UNIX とするものと，(2) HostOS を UNIX，GuestOS を Windows とするものが考えられる．

いずれの構成においても，クライアント側では 2 種類の OS が稼働するため，ファイルサービスの提供方法に問題が生じる．すなわち，サーバ側でそれぞれの OS に対応したファイルサーバを準備する場合には，サーバが 2 種類となり，管理コストが上昇する．

もしくは，両方の OS で共通のファイルサーバを利用する場合においても，一般には複数のファイル共有プロトコルを提供する負荷が生じる．特に複数の OS から共通のファイルを同時に変更したり，一方の OS が利用しているディレクトリを削除したりしてしまうことがないようにするために，排他処理に注意を払う必要が生じる☆．

このほか，クライアント側で 2 つの OS が同時に稼働することとなるため，システムのリソースに負荷をかけることとなり，電源投入から利用可能となるまでに時間がかかる問題もある．特に，GuestOS は HostOS の起動後に起動されるため，GuestOS が利用可能となるまでには相当の時間を要する．教育用システムとしてのユーザビリティを下げないためには，この起動時間の短縮も必要となる．

以下，これらの問題点についてより詳細に検討するとともに，従来の解決法について示す．

2.3 Virtual Machine を用いたシステムに対するファイルサービス

大規模な計算機システムのためのファイルサーバは多数の端末への効果的なサービスを実現することが求められる．

ファイルサーバを総ディスク容量やクライアント数の面でスケールさせる素直な構成法としては，サーバを複数台並列に並べる手法が考えられるが，その際，システム運用の管理コストを下げるためには，サーバを 1 台として扱えること，すなわち SSI（Single System Image）となっていることが重要である．また，複数のサーバによる構成では，ユーザとサーバの対応付けを何らかのディレクトリサービスを用いて管理する必要が生じるため，特に大規模なシステムでは管理

☆ 特に作業ファイルやそれの置かれているディレクトリを削除するような操作は，頻度は少ないとしてもその影響は重大なものとなるため，適切な排他処理を行えることが必要である．

コストが増える要因となる．そこで本論文ではサーバを複数台にすることなく運用できるシステムを前提とした．

さらに，本論文が対象とする 2 種類の OS が稼働するシステムでは，ホームディレクトリをどのように構築するかがさらに問題を複雑にする．従来は Windows と UNIX のそれぞれにホームディレクトリが存在し，それらを交互に利用していたために，両 OS 間でのファイル操作は FTP 等を利用した転送を行う必要があった．この作業は手間がかかるだけでなく，同じデータがホームディレクトリ上に複数記録され，ディスクの無駄を生じる原因となっていた．

それに対し Virtual Machine を使ったシステムでは，利用者は Windows と UNIX の両方を同時に利用することも想定されるため，従来のホームディレクトリ構築法では OS 間でのファイル複製頻度がさらに増え，手間とディスクの無駄を助長する．

この手間と無駄を削減するためには，2 つの OS 間でホームディレクトリを共有することが望まれる．しかし，これら 2 つの OS は，ファイル共有で利用するプロトコルが異なるため，ホームディレクトリの設計に工夫が必要となる．

Windows ではファイル共有において SMB プロトコルが用いられる．このプロトコルはステートを持つプロトコルのため，クライアントごとのステートをサーバ側で把握する必要がある．そのため，クライアントの台数が 1,000 台を超すような大規模システムでは，管理すべきステートが膨大な量となり運用は困難となる．また，ステートの存在は，障害時のファイルサーバの再起動が利用者全体に対して影響を与えかねない．

一方，UNIX ではファイル共有において NFS プロトコルを用いる．このプロトコルはステートレスなため，大規模なシステムの場合にも運用しやすく，クライアント数が 1,000 台を超す大規模システムでの実績もある．

また，NFS はステートレスなプロトコルであるため，ファイルサービスの面においては，障害発生時にはサーバを再起動するだけでクライアントに影響を出さずに復旧できる．

しかし，ファイルアクセスにおける排他制御（ロック）について検討すると，NFS 自体にはロックの機構はなく，通常は NLM [9] に従うロック機構を利用する．すなわち，NFS を利用する場合でも SMB の場合と同じく，サーバ上でロック管理を行うことになるため，サーバがステートを持つことになってしまい，結果として大規模な運用に適さない．

表 1 NFS と SMB の比較
Table 1 Comparison between NFS and SMB.

	NFS	SMB
UNIX との親和性	◎	○
Windows との親和性	×	◎
大規模運用での実績	◎	△
接続の管理	Stateless	Stateful

これまでに述べた NFS と SMB の特徴の比較を**表 1** にまとめる．

一般に SMB と NFS とを同時に利用し，かつ，ホームディレクトリを共有するシステムを構築する際には，(1) NFS サーバで Samba [10] 等のアプリケーションを利用して，SMB プロトコルによるファイル共有も行えるようにする，(2) 両方のプロトコルをサポートする NAS（Network Attached Storage）を利用する，(3) Windows サーバ上で Service for UNIX [11]（SFU）を稼働させ，NFS もサポートするようにする，(4) Windows クライアント上で Service for UNIX を稼働させ，NFS サーバに接続するようにする，といった手法が考えられる．以下，その得失を検討する．

2.3.1 Samba を用いる方式

UNIX 上に NFS サーバを稼働させたうえで，Samba 等☆を利用して SMB プロトコルによるサービスも行う方式が候補にあがる．しかし，SMB プロトコルはステートを持つプロトコルであるため，サーバ上にクライアント数に比例したステート情報が必要となり，大規模な運用には適さない．SMB プロトコルの実装として Samba を例にあげると，クライアントが 300〜500 台を超えると実用的ではなくなることが確認されている☆☆．

2.3.2 NAS を用いる方式

SMB と NFS の両方のプロトコルをサポートする NAS は多数存在するが，その多くの内部構造は，UNIX ベースのファイルシステムの上で Samba ないしは同等のプログラムを利用しており，いずれも上述した問題点を含んでいる．Windows ベースの NAS も発売されつつあるが，サーバ自体にパッチの適用が必要と

☆ UNIX 上で SMB プロトコルをサポートするプログラムは Samba 以外にもいくつかあるが，Samba が事実上の標準となっている．

☆☆ 文献 12) によると，Samba が消費するリソースは n をクライアント数として，
smbd が消費する File Descriptor 数=23 $\times n$
smbd が消費するメモリ（MB）$=1.1 + 290/1024 \times n$
である．また，Samba の設計上の最大クライアント数は 3000 と示されているが，この試験においても，同時に 300 クライアントからの接続しか試験されていない．

なるといった点で安定性に欠けるため，大規模な運用に耐えられない．

2.3.3 Service for UNIX をサーバで用いる方式

Samba とは逆に，Windows サーバ上に SMB サーバを構築したうえで，そのサーバ上で SFU に含まれる NFS server を利用することも検討できる．しかし，この方式は Windows サーバ自体のセキュリティを維持するために，システムの更新時に再起動が必要となることが多く，ファイルサーバとしての運用には困難がともなう．

2.3.4 Service for UNIX をクライアントで用いる方式

2.3.3 項と逆に，UNIX サーバ上に NFS サーバを構築して，Windows クライアント側で SFU を利用することで，Windows クライアントを NFS サーバに接続することも検討できる．NFS プロトコルの利用によりステートを管理する必要がなくなる．

しかし SFU では，NFS が利用するユーザ ID と SMB が利用する Windows アカウントの対応付けを，SFU に附属する User Mapping Server を用いて管理する必要がある．このため，User Mapping Server を管理するコストが増えるだけでなく，すべてのクライアントがこのサーバを参照するため，単一点障害で全クライアントに影響が出てしまう☆．また，SFU では NLMv4[9] に従うファイルロックを利用することもできるが，この場合はサーバ側でステートを管理する必要が生じてしまう．

2.4 Virtual Machine を用いたシステムの起動時間の短縮を考慮した更新手法

Virtual Machine を利用したシステムは，CPU やメモリといったリソースを HostOS と GuestOS とで分けあうことになるため，あらゆる処理において通常よりも時間がかかる．

特に電源を投入してから OS が起動するまでの時間は，Virtual Machine を使うか否かにかかわらず，ユーザにとっては待ち時間であるため，この時間を短縮することはユーザビリティの観点からは非常に重要である．すなわち，Virtual Machine を利用したことで，この待ち時間が 2 倍となってしまうことがないように工夫をする必要がある．

ところで電源投入後 OS が利用できるようになるまでの時間を短縮するために，ハイバネーション技術が利用できる．ハイバネーションを利用することで，OS が起動した後での CPU やハードディスクの状態を記録し，後にその状態から OS の利用を再開することができる．

この手法を Virtual Machine 内で稼働する GuestOS に対して利用することで，Virtual Machine を起動してから GuestOS が利用可能になるまでの時間を大幅に短縮できる☆☆．さらには，ハイバネーションを行った際に作成される CPU やハードディスクの状態を繰り返し利用することで，利用者に毎回同じ状態からのサービスを提供することが可能となり，システムの安定運用を行ううえでの役割は大きい．

ところが，この手法を利用して毎回同じ状態から端末を起動すると，セキュリティ問題や設定の変更といった GuestOS のファイルの修正等をいつどのような手順で行うのかという点に困難が生じる．

まず，このような修正は Virtual Machine がハイバネーションから復帰してユーザが利用を開始する前に行う必要があるが，GuestOS はそれ自身ではいつ Virtual Machine がハイバネーションから復帰したのかを知ることはできないため，パッチを適用する処理をいつ行えばよいのかを知ることは困難である．さらには，パッチを適用したあとに再起動が必要となると，そのシステムのユーザビリティは著しく低下することとなる．

もちろん，パッチを適用したあとに再度ハイバネーションイメージを作成しなおすことも検討できる．しかし，パッチ適用後のイメージの作成は，GuestOS をユーザが利用していない時間帯に行う必要がある．しかしながら 24 時間運用を前提とした教育用システムでは，このような作業を合理的な手順で行うことは非常に困難である．

このような修正の手間を減らすために，GuestOS が利用するファイルの大半をネットワーク越しに共有することも考えられる．特に，アプリケーションや OS の設定ファイル等は，ファイルサーバ上の共有領域に置かれている方が，ファイルの更新の容易さの点からは有利である．しかしながら，教育用計算機システムでは，講義等における指示によって，多くのクライアントでいっせいに同一アプリケーションを起動することがめずらしくない．そのため，ファイルサーバの負荷を考慮すると，ファイルサーバ上にこれらのファイルを置く手法は採用しにくい．

☆ 本論文執筆時の SFU は Ver 3.0 であった．最近発表された SFU ver3.5 において，“Mapping Server Redundancy” の機能が追加され，Mapping Server を複数のサーバ上で同時に動作させることが可能になった．

☆☆ HostOS に対してもハイバネーションを利用することで，HostOS の起動後ユーザが利用できるようになるまでの時間を短縮することができるが，これは本論文の議論の範囲とはしない．

そこで GuestOS のファイルを更新する際には，GuestOS の個々のファイルを変更するのではなく，ファイルシステム全体を置き換えるという手法がとられることも多い．しかし，先の理由により GuestOS が利用するファイルの大半は，ファイルサーバ上での共有に適さないため，GuestOS のファイルシステムのサイズは大きなものとなる．

文献 6) では，GuestOS のファイルシステム全体をネットワーク越しに転送して GuestOS の更新を行っている．しかしながら GuestOS のファイルに対するわずかな変更を行うだけでも，ファイルシステム全体を転送する必要がある．

同文献では GuestOS として Windows を選んでいるが，転送量は 3 GB におよび，20 台の端末への転送に約 1 時間を必要とした．また，GuestOS の修正には，セキュリティ問題の修正のためのもののように，即時の適用を必要とする修正が多いため，「バックグラウンドで時間をかけて取得する」，「修正プログラムを分割して，徐々に配信する」といった時間のかかる方法も適切ではない．すなわち，この方式はシステムの規模に対するスケーラビリティに欠け，文献 6) のように端末数が 250 台の環境では問題ないが，1,000 台を超す環境では実用的ではなくなる．

さらには，HostOS のファイルシステム全体を更新する手法により，GuestOS のファイルシステムを更新することも検討できる．HostOS のファイルシステム全体の更新は一般にディスク全体の更新を意味するが，この作業は通常は同一セグメントに含まれる複数台の端末をいっせいに，サーバからのマルチキャスト通信を用いて行うことになる．

しかしながら，この手法は転送するべき量が膨大となることから，作業に時間がかかるだけでなく，配信後に端末ごとの個別設定を行う必要があるため手順が複雑となりやすく，日常的にこの手法を利用して GuestOS の修正を行うことは困難である．

3. 教育用計算機システムに適したホームディレクトリ提供手法

本章では Virtual Machine を用いた教育用計算機システムのホームディレクトリとしての利用に適したファイルサービス提供手法を提案する．

Windows と UNIX の両方からアクセスできるファイルシステムについては 2.3 節において議論をしたが，いずれの手法もサーバ側において集約的に管理する情報を持つために，大規模な運用には適さない．

表 2 クライアントで使うファイルの分類
Table 2 Classification of files of remote host.

		ローカルになければならないか	
		YES	NO
複数のクライアントから同時に読み書きするか	YES	—	Type A
	NO	Type B	Type C

Type A：共有データ（講義資料等）
Type B：OS 本体，アプリケーション，一時ファイル
Type C：ホームディレクトリ

3.1 端末が利用するファイルの分類

サーバとクライアントの間でのファイル共有の検討にあたり，まずクライアントが利用するファイルを，(1) ローカルになければならないかどうか，(2) 複数のクライアントから同時に読み書きできる必要があるかどうか，という観点で**表 2** のように分類した．

各利用者ごとのホームディレクトリは Type C に分類され，サーバとクライアントとの間で共有されるものである．教育用計算機システムにおけるホームディレクトリには

- 同一ユーザによる複数の端末の利用は，特に配慮する必要はない，
- 同一ファイルを同一クライアント内の Windows および UNIX から同時にアクセスすることを想定して，ロックが働くようにしなければならない，

といった特徴がある．

すなわち，ホームディレクトリの領域にアクセスする場合には，ファイルやフォルダのロックといった排他処理を異なるクライアント端末間で行う必要はなく，各クライアント内の Windows と UNIX との間における排他処理のみを考えればよいことになる．

なお，同一ユーザが複数の端末から同時にログインする，いわゆる「重複ログイン」はないものとする．重複ログインを禁止する手法については，3.5 節で議論する．

3.2 提案手法の詳細

本節では，クライアント端末の UNIX において，NFS から SMB へのプロトコル変換を行い，利用者のホームディレクトリに関するファイルロック等のステート情報をクライアントの UNIX 側で一元管理する手法を提案する．

以下に，提案手法を，ファイルサーバからクライアントまで順に述べる．

(1) ファイルサーバでは各ユーザのホームディレクトリを NFS によりサービスする．NFS サーバでは NFS 越しの lock[9] を利用できるようにする必要はない．

(2) クライアントの UNIX において，ホームディレクトリを NFS マウントする．マウントしたホームディレクトリ内での Lock のリクエストは NLM を用いてサーバに lock 処理を転送するのではなく，クライアント UNIX 内で管理する．具体的には，クライアント内の open，lock の両システムコールを変更し，ホームディレクトリ領域に対する lock 要求を samba と同じ手法で管理するように変更する☆．

(3) マウントしたホームディレクトリを SMB プロトコルで再度 export する．SMB クライアントからの Lock リクエストは，上述と同じ方式で管理する．

(4) 端末の Windows は，同じ端末の UNIX 上にあるホームディレクトリを SMB で共有する．

この手法でのホームディレクトリの共有により，

- サーバは UNIX で構成されるため，セキュリティ問題等の修正を行った場合においても，再起動が必要となる頻度は低い，
- サーバは NFS でサービスを行い，ステート情報を持たない．このためクライアント数が増えた場合においても，接続数そのものの増加による直接的な影響でサーバの負荷が増えることはない，
- ユーザのホームディレクトリ内のファイルに対するロック情報は，各ユーザが利用するクライアント内の Windows と UNIX とで共有されつつ一元管理されるため，クライアント上で稼働する 2 つの OS から同じファイルにアクセスがあったとしても問題にはならない，
- SMB プロトコルのステート情報はクライアント端末内のみで管理されるため，ステート情報に何らかの問題が生じたとしても，その影響がファイルサーバには及ばず，端末の再起動により復旧できる，

といった特徴を備えることになる．

なお，この手法を用いるうえでは，

- SMB と NFS との間でプロトコル変換を行うためには，SMB 側でのユーザ名と NFS 側でのユーザ ID との間での対応付けが必要となる，
- SMB による再 export が行われるまで，Windows 側からホームディレクトリを利用することはできない，
- 重複ログインをしたときにロックが働かない，

といった問題が生じるが，これらの問題については次節以降で順次述べる．

3.3 ユーザ ID のマッピング

UNIX と Windows の両方を利用するシステムにおいて，利用者の利便性を考えるとユーザ ID とパスワードは Windows 側と UNIX 側とで共通にすることが好ましい．一般に，クライアント側にユーザ情報を保持することは好ましくないため，これらの情報は何らかのディレクトリサービス上に保存しておく．

また，クライアント側で NFS から SMB へのプロトコル変換を行うためには，上記の情報に加えて「UNIX 側で内部的に利用する UID」および，「SMB のパスワードハッシュを生成するための生パスワード」が必要となるため，これらもあわせてディレクトリサービス上に保存しておく．

3.4 認証時の問題

一般に GuestOS は，HostOS 側での認証の後に起動される．HostOS 側で認証を行う前から，あらかじめ Virtual Machine を起動しておくことも検討はできるが，セキュリティを犠牲にすることなく，認証後の HostOS 側の画面に Virtual Machine の窓を表示させることは困難であり，GuestOS の画面をユーザが直接操作することができなくなり，利用上の不便が生じる．

そこで GuestOS は HostOS 側での認証を行った後で起動するようにすると，今度は GuestOS と HostOS との間でシステム上の依存関係がある場合，特に GuestOS 側で UNIX を稼働させる場合には，Windows の認証時後の動作に問題が生じる．

Windows はその仕様上，GINA[14] によるログオン認証の直後に，ホームディレクトリへのアクセスが可能かどうかを調査する．この調査において，「サーバが起動されていない」ないしは「サーバとの接続が遅い」と判定された場合には，管理用スクリプトが実行されない等の不具合が生じることになる[13]．

ところが，Samba を GuestOS 内で起動する場合や，HostOS で起動する場合でもあらかじめ起動しておけない事情がある場合等においては，認証の完了までに samba の設定と起動を完了することは困難である．

このような場合には，(1) Windows のレジストリを変更して，低速リンクの検出待ち時間を変更，ないしは検出を行わないようにする，(2) GINA[14] を拡張して認証直後にスクリプトを実行できるようにログオン処理の流れを変更する，のいずれかを行う必要がある．

3.5 重複ログイン時にロックが機能しない問題

これまでの議論は，ユーザごとに見ると，ホーム

☆ samba が利用している locking.tdb というデータベースを UNIX の lock も利用して，両者でロック情報を共有して管理することになる．

ディレクトリへの複数の端末からの同時アクセスがないことを前提としている．そのため，同一の利用者が複数のクライアントを同時に利用して，同じファイルに同時にアクセスしたり，アクセス中のディレクトリやファイルを削除したりすると，ファイルロックが正常に働かない．

この問題は本来は運用ポリシにより禁止されるべきものであるが，必要に応じてログイン認証の際にログインの唯一性のチェックにより対応することもできる．

4. GuestOS の起動時間の短縮

HostOS および GuestOS へのセキュリティパッチの適用や設定変更（以下,「パッチ等の適用」とする）の影響について検討する．

HostOS 側の OS については，一般的な物理ディスクを使っている場合にはパッチ等の適用には特に問題は生じない☆．また，パッチの適用後に再起動が生じたとしても，その処理は一度限りであり，かつ一般にはユーザがログインする前に処理を完了できるため，大きな問題とはならない．

それに対して，GuestOS の起動時間の短縮のためにハイバネーション技術や Repeatable Resume 技術を利用することが有効であることを 2.4 節において述べた．しかしながら GuestOS に対してパッチ等を適用行った際に再起動が必要となると，起動時間の短縮の目的が達せられないことになる．

UNIX ではこのようなパッチ等の適用にともなって再起動が必要となることはめずらしいが，Windows では適用するセキュリティパッチには再起動を必要とするものが多い．この点からは，GuestOS の選択として UNIX が適切であると考える．

4.1 GuestOS へのパッチ等の適用手順

GuestOS の起動（復帰）に際してパッチ等の適用の終了まで利用者がログインしないように設定する必要がある．Virtual Machine の起動はユーザのログイン操作後であるため，パッチ等の適用に時間がかかれば，その間，利用者が待たされることとなる．したがって，GuestOS の起動時間だけでなく，パッチ等の適用に要する時間を短縮する必要もある．

また，2.4 節で述べたが，起動時間の短縮にはハイバネーションの利用が効果的である．また，GuestOS を安定に運用するためには，ユーザが GuestOS を利用するたびに，GuestOS のディスクイメージを初期化し，毎回同じ設定に戻すことが好ましい．そこでハイバネーションにあわせて 2.1 節で述べた Repeatable Resume を利用することになる．

しかし，Repeatable Resume を行うと，GuestOS にセキュリティパッチ等の修正プログラムを適用する前のハイバネーションを実行したときの状態に戻されてしまう．したがって，GuestOS がハイバネーションから復帰した際に毎回パッチ等を適用しなおす必要があるが，パッチ等を毎回ダウンロードするのは，非効率なだけでなく，サーバの負荷や転送時間の面からも実用的ではない．

この問題の解決策として，GuestOS の起動から利用者がログイン可能となるまでの手順を次のように設計した．

(1) HostOS のディスク内に，GuestOS からもアクセスできる領域を作る．
(2) HostOS 起動時に，GuestOS 用の新しいパッチ等が増えているかどうかを確認し，必要に応じてダウンロードする．
(3) HostOS 側に利用者がログオンする．
(4) Virtual Machine を起動して，GuestOS をハイバネーションから復帰させる．
(5) HostOS 上にダウンロードされた修正プログラムを，順次適用する．
(6) ユーザのログインを許可する．

ここで述べたパッチ適用の手順のうち，個別のパッチに対して処理時間が必要となるのは次の 3 つの段階である．

(a) 新しいパッチが増えている際に，サーバからクライアントへパッチを転送する処理
(b) HostOS 側のパッチの置かれたフォルダを，GuestOS からアクセスできるようにする処理
(c) GuestOS 内で，順次パッチを適用する処理

これらのうち，(a) の処理は端末の起動時ないしはハイバネーションからの復帰時に行われる．この処理に必要な時間は，パッチファイルをサーバから転送する際に必要な時間である．パッチファイルのサイズが大きいときには転送処理に時間がかかることも予想される．しかし，この転送処理はユーザのログオン手続きとは独立かつ並行して行えるため，そのように設定することでたとえ転送に時間がかかったとしても利用者にとって負担とはならない．

次に (b) の処理時間であるが，mount 処理において数秒程度の時間が必要となる場合が生じるかもしれないが，この間に行われる処理は各端末で独立して行わ

☆ HostOS 側の物理ディスクに対してディスク復元ツールを使っている場合には GuestOS に対して Repeatable Resume を用いている場合と似た問題が生じるが，この点は本論文での議論の対象外とする．

れるものであるため，複数の端末で同時に行っても処理時間が延びるものではなく，また，この処理時間は周囲の状況に影響されることもなくそれほど変動しない．

最後に (c) の処理だが，単純なファイルのコピー作業の場合には，実質問題とならない程度の時間で処理が完了するものと思われる．もし，ファイルの比較やプログラムの実行等でこの処理に時間がかかるものの場合においても，バックグランド実行が可能なものであれば大きな問題にはならない．ただし，セキュリティ問題の修正等，利用者がログインするまでに修正する必要があるときには，修正が完了するまでログイン処理に移らないように待つ必要がある．すなわち，利用者に直接的に影響するのは，パッチ作業に時間のかかる処理で，かつ，パッチ処理の終了までログインを認めてはならないときである．

なお，この手法は，GuestOS 用のパッチ等のサイズがそれほど大きくなく，かつ，その適用に際して再起動を必要としないことを前提としている．通常 UNIX に対して適用するセキュリティパッチ等はこの条件を満たす．ただし，この手法が適用できない場合には，2.4 節で述べたように，ディスクのイメージファイル全体の入れ替えが必要となる．しかしながら，端末ごとに個別にディスクイメージを入れ替える作業は，大規模なシステムでは実用的ではない．そこで，HostOS も含めたディスクイメージ全体のマルチキャスト配信により更新する際に，同時に GuestOS も含めて配信することになる☆．

5. 提案手法の適用

5.1 構築されたシステムの概要

平成 13 年度に行われた京都大学学術情報メディアセンターの教育用計算機システムの更新にあたり，本提案手法を適用した[15),16)]．構築されたシステムは約 1,200 台の利用者端末を学内 30 カ所以上のサテライト教室に分散配置するという大規模なものである．概要図を**図 1** に，ファイルサーバおよび端末の性能を**表 3** に示す．

Virtual Machine を利用してすべての端末で Windows と UNIX の両方を利用できるようにした．Virtual Machine は，VMware 社の VMware Workstation Ver3.2[8)] を用いた．

端末の構成法として Windows と UNIX のどちらを HostOS にするかの検討にあたっては，4 章で述べた理由に加えて，GuestOS 側では画面描画の速度が遅くなる点を重視した．

利用の実態を分析したところ，Windows においては CAD 等の高速な画面描画を要求するアプリケーションが利用されていたが，UNIX において同様の要望はなかった．そのため，今回の構築においては，Windows2000 を HostOS，UNIX を GuestOS として構築することにした．

物理メモリ（256 MB）のうち Virtual Machine に 128 MB を GuestOS 用として割り当てた．GuestOS となる UNIX 環境は，VMware 上での安定動作の点から Linux を選択し，各種アプリケーションの点からディストリビューションとして Vine Linux Ver2.1.5[17)] を選択した．

ハードディスクは 20 GB のうち最大 10 GB を Virtual Machine 内の Linux が利用できるように設定した☆☆．Virtual Machine のハードディスクは Non Persistent Mode に設定した．また HostOS 側の起動時間を短縮するために，HostOS のハイバネーション技

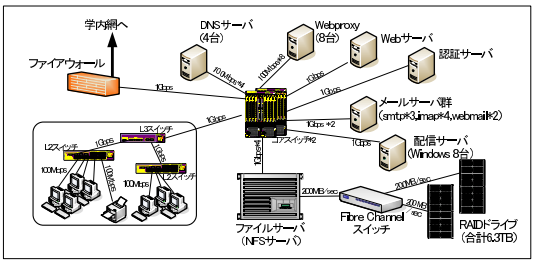


図 1 システム構成の概要
Fig. 1 Overview of the system.

表 3 ファイルサーバと端末の性能
Table 3 Specifications of the system.

	課目	性能
NFS サーバ	CPU	PA-8600 (550 MHz) × 4
	Memory	4 GB
	Specint_rate95	1468
	OS	HP-UX11i（64 bit）
	NIC	1000BASE-T × 4
RAID ドライブ	Host I/F	Fibre Channel
	ディスク容量	6.3 TB
	搭載ディスクドライブ	71.6 GB/DISK
	搭載キャッシュ容量	2048 MB/コントローラ
利用者端末	CPU	PentiumIII 1 GHz
	Memory	256 MB
	HDD	20 GB
	NIC	100BASE-TX

☆ 端末のディスクイメージ全体の更新を必要とするような，大規模な設定変更やソフトウェア追加等の作業は，講義のスケジュールにあわせて，少なくとも年 2 回以上必要である．

☆☆ VMware は GuestOS が消費したディスク量に応じてディスクを消費する．現在のところ 2.3 GB を消費している．

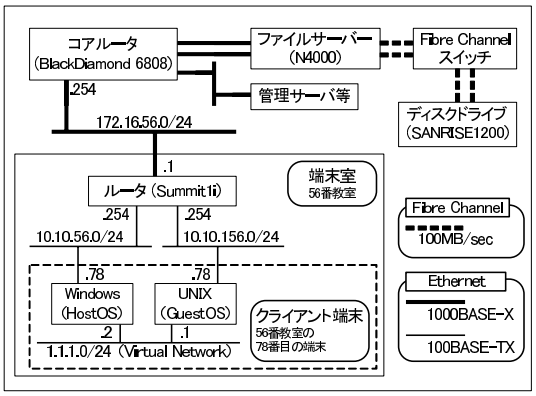


図 2 端末周辺のネットワーク構成図
Fig. 2 Network structure of the system.

術も用いることにした.

5.2 ネットワーク構成

クライアント内でのネットワーク構成は，次のようにした.

(1) GuestOS の管理を管理サーバからネットワーク越しに直接行えるように GuestOS にブリッジモードのインタフェースを持たせ，IP アドレスを割り当てる.

(2) 盗聴される心配なく管理情報を交換できるように，Virtual Network モードのインタフェースを持たせてクライアント端末の Windows と UNIX の間に安全なコネクションを作る.

(3) 各端末の設定と OS 間での通信を容易にするため，Virtual Network で利用する IP アドレスをすべての端末で同じにする.

構築されたネットワーク構成図を**図 2** に示す．各端末にはネットワークインタフェースは 1 つずつしか存在しないがこのような構成を利用するために，同一の物理線の上に 2 つのネットワークセグメントを流すことにした.

5.3 Virtual Machine に関する詳細な設定

5.3.1 ユーザがログオンするまでの詳細な流れ

UNIX は Virtual Machine 上で動作するため，UNIX を利用する際には，まず Windows にログオンした後で再度 Linux にログインする必要がある．このため，Windows のログオン処理において，同時に Linux 側においても X サーバを起動し，ログインしたユーザの権限でシェルを起動してログイン済みの状態となるようにした☆1.

以下に，ユーザが Windows にログオンしてから Linux を利用できるようになるまでの詳細な流れを順に示す.

(1) ユーザが Windows にログオンする.

(2) Virtual Machine を起動し，Linux をハイバネーションから復帰する.

(3) Windows はユーザ情報サーバに ssh を用いて接続し，ログオンしたユーザ名を受け渡す.

(4) ユーザ情報サーバは渡されたユーザ名を元にユーザ ID とパスワード☆2を取得し，それらを ssh を経由して Linux に受け渡す.

(5) Linux は ssh で接続されたことを契機にパッチの適用等の処理を行う.

(6) Linux は渡されたユーザ名とユーザ ID を用いて UNIX アカウントを作成すると同時に，ユーザ名とパスワードを用いて Samba の設定を行う．処理が完了したら Windows 側に通知する.

(7) Windows は Linux 完了の通知を受けて，ホームディレクトリの準備ができたことを確認してから，ログイン処理を継続する.

(8) Linux 側で X Window を起動し，ユーザ権限で shell を立ち上げる.

ここに示すユーザ情報サーバにより，2.3.4 項において述べたユーザ ID と Windows の対応付けを行っている．現在はすべて独自の実装によるものであるが，将来的には LDAP 等のディレクトリを参照するように変更することも容易である．また負荷状況によっては，同等のサーバを複数台設置することも容易である.

また，GINA [14] を独自に作成したものに変更して，Windows のログオン画面に，Windows と Linux のどちらを利用したいかを選択するためのチェックボックスを設けた☆3．Linux が指定されたときには Windows のログオン処理のあと Virtual Machine のウィンドウを自動的に全画面表示にするようにした.

逆に Windows が指定された際には，Virtual Machine のウィンドウを意識することがないように画面上から隠すようにした☆4．このように設定することで，

☆1 Linux 環境はシェルは tcsh，ウィンドウマネージャは twm を標準として提供している.

☆2 Samba を利用するうえでは，ユーザの生パスワードが必要となる.

☆3 今回の実装においては，GINA の変更をすることで Linux を全画面で起動することもできるようにしたが，もし GINA を変更しないとしても，GuestOS の起動手順等，本論文の課題である「Virtual Machine を活用したシステム構築技術」には大きな影響はない.

☆4 隠された Virtual Machine のウィンドウは，スタートメニューから選択できるツールを利用することで，再表示することができる.

Windows のみを利用するユーザ（このようなユーザの割合が最も多く，かつ初心者が多く含まれることが想定される）に不必要な抵抗感が生じるのを防ぐようにした．

3.4 節で述べた問題には，上述のとおり GINA を入れ替えることで対応した．

5.3.2 Repeatable Resume の高速化

Virtual Machine は Repeatable Resume を利用するように設定した．VMware の実装における Repeatable Resume の処理では，ディスクの復元処理と，メモリのイメージとハードウェアの構成情報の復元処理の大きく 2 つの処理が行われる．

ディスクについては，ある時点でスナップショットを作成して以降のディスクへの変更は別ファイル（ディスクイメージ差分ファイル）に書き込まれているため，復元処理においてはディスクイメージ差分ファイルを削除するだけでよい．

それに対して，メモリのイメージとハードウェアの構成情報はスナップショットを作成した段階のデータがバックアップファイルとして保存されているため，復元処理においてはこれらのファイルを複製することになる☆．

複製されるファイルはメモリイメージの情報をそのまま含むため，メモリとほぼ同等のサイズのファイルになる．今回構築したシステム構成では約 135 MB のファイルサイズとなったが，このファイルの復元には平均 23.2 秒を要した．

そこで VMware の Repeatable Resume 機能をそのまま利用するのではなく，同等の処理を行うように独自に実装することにした．また，文献 6) と同じく複製処理をログオフ時に行うようにしたうえで，単に複製ではなく，実行形式に圧縮されたファイルを展開するように工夫した☆☆．これにより復元処理に要する時間は平均 12.2 秒となり，約 11 秒の短縮ができた．

5.3.3 セキュリティ対策

一般ユーザが勝手に新しい Virtual Machine を作成するのを防ぐため，Virtual Machine は利用者制限を行うためのラッパを経由しないと起動できないようにしている．

また，Virtual Machine が停止してしまうと，ファイルシステムの利用ができなくなるため，Virtual Machine の停止を防ぐ設定もされている．

5.3.4 Linux に対するパッチ等の適用

GuestOS である Linux に対しては，HostOS である Windows の起動時およびハイバネーションからの復帰時に，4 章で述べた手順によりパッチ等の適用をすることにした．

追記すべき点としては，

(1) HostOS である Windows 内にサーバからパッチをダウンロードして保存するフォルダを作成し，そのフォルダに対する Windows 共有を許可する，
(2) HostOS の起動時およびハイバネーションからの復元時に，Web サーバを確認し，新しいパッチ等が存在している場合には上記フォルダにダウンロードする，
(3) HostOS 側にユーザがログオンし，GuestOS がハイバネーションから復帰したときに，上記 Windows 側のフォルダを smbfs を用いて mount する，
(4) mount したフォルダ内に，GuestOS に対して未適用のパッチが存在する際には，そのパッチを古いものから順に適用する，

があげられる．

それぞれのパッチはバージョン番号を示すフォルダ内に必要なファイルが配置され，そのフォルダ内の patch.sh というシェルスクリプトを起動するとパッチが適用されるようにすることで，パッチ適用手順の統一化を図っている．

5.4 NFS サーバと SMB サーバの設定

最大約 3 万人と想定された利用者数に対して同時に平均 100 MB の容量を提供できるように，3 TB の容量を持つファイルサーバを導入した．

ファイルサーバには日立製 N4000（HP-UX11i）に SANRISE1200 を Fibre channel 経由でミラー構成で 2 台接続したものを用いた．また，NFS のバージョンとトランスポート層のプロトコルの組合せについては転送速度を調査した結果，TCP，NFSv3 の組合せを利用することにした．

Windows におけるホームディレクトリは 3 章で述べた手順を用いて構築された．この構築にあたり SMB プロトコルのサーバには，問題が発生した際に問題点の解析の容易性を考慮して，オープンソースで開発されている Samba を用いることにした．

☆ メモリイメージも差分のみを保存する手法も検討されるべきだが，少なくとも VMware Workstation Version 4.5 においても，この挙動は変更されていない．

☆☆ 実行形式にすることで，展開時にはいったん全体をメモリ上に「全体を読み込んだうえで書き込みが行われる」ため，ディスクに対しては連続したアクセスが行われることが期待される．それに対して解凍ツールを利用するとディスクから「読み込みながら書き込む」動作を繰り返すことになるため，結果として速度に差が生じる．

なお，Windows と Linux との間での SMB プロトコルによるファイル共有は非常に高速な上述の Virtual Network の上で行われるため，この区間での速度低下や遅延はほとんどない．

なお，NFS サーバ上に，2 バイト文字を含むファイルを作成すると，サーバの運用が困難になることがある．このような問題を避けるために Samba の設定にあたっては，利用者のクライアントからどのような文字コードでアクセスがあった場合においても，NFS サーバ上に 2 バイト文字を含むファイルを作ることがないように配慮した☆．

6. 提案手法の評価

3 章および 4 章において提案した手法の効果を確認するために，5 章で構築したシステムについて評価を行った．

6.1 プロトコル変換のオーバヘッドとファイル読み書きの速度

3 章で提案した手法では，プロトコル変換にともなうオーバヘッドが問題となりうる．また，Windows からファイルアクセスをする際には，それぞれのプロトコルで利用するバッファサイズの影響を受けることになる．そこで，これらの影響を見積もるために，

- 提案方式（SMB を NFS に変換してファイルサーバにアクセス），
- Linux から NFS で直接アクセスした場合，
- サーバに Samba を導入して Windows から直接 SMB でアクセスした場合，

の環境をそれぞれ構築し，読み込みおよび書き込みの速度を測定した．

データのサイズが小さいときには CPU やメモリ上のキャッシュにヒットする可能性があるため，1 度に読み書きするデータサイズを 64 KB に固定したうえで，64 KB から 128 MB までの各サイズのファイルを転送した際の転送速度を測定した．測定にはハードディスクおよび NFS のベンチマークとしては標準的に利用されていて，かつ Windows と Linux の両方で利用できる iozone[18] を用いた．測定は 2 台の PC（PC1, PC2）でそれぞれ 10 回ずつ行い，平均値および標準偏差を測定した．

結果を，横軸をファイルサイズ，縦軸を転送速度として，**図 3** および **図 4** に示す．

測定結果では，提案方式でのファイルアクセスは読み込みでは 7.3 MByte/sec 程度，書き込みでは 1.5 MByte/sec 程度の速度となった☆☆．この速度は，読み込みについては，NFS や SMB のみによるアクセスに対して遜色なく，十分実用に耐えると考えられる．書き込みは従来の方式と比して遅いが，一般的な利用法で書き込みが頻繁に発生することは少ないため，大きな問題とはならない．

ただし，プログラミング教育等においてコンパイルを行う際には，書き込みが頻繁に発生することが予想される．これに対しては，ローカルディスク上でコンパイル処理を行う等の運用上の工夫が有効であると思われる．

また，読み込みにおいて，読み込みサイズが 10 MByte 付近より速度低下が見られるが，これは主

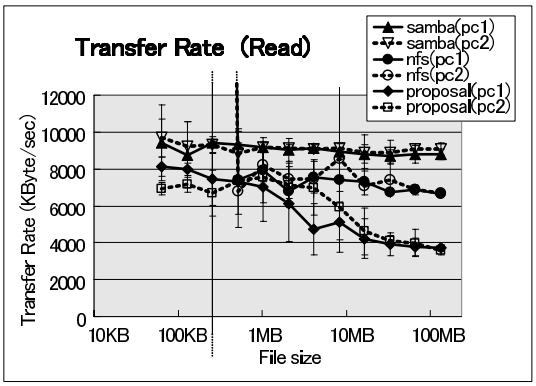


図 3 ファイルサイズによる読み込み速度の変動
Fig. 3 Speed measurement of file system according to the size of file (Read).

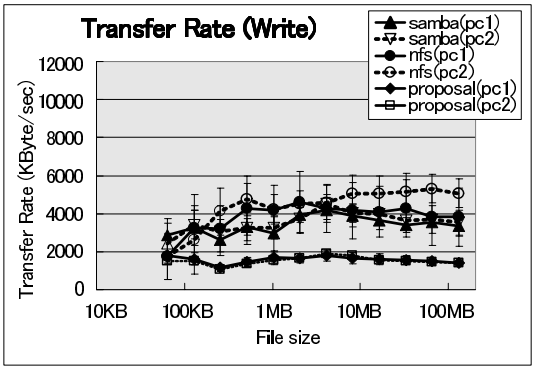


図 4 ファイルサイズによる書き込み速度の変動
Fig. 4 Speed measurement of file system according to the size of file (Write).

☆ この点が利用者にとって問題とならないように，UNIX 上のファイル名を 2 バイト文字を含む表現に戻すためのフィルタを利用者に提供している．

☆☆ 文献 16) と測定結果が異なるが，これは主に測定ツールを変更した影響であると考えられる．

に物理メモリの量による影響と考えられる．しかし，利用者のホームディレクトリに対しては容量制限☆を課しているため，利用者が大きなサイズのファイルを扱うことは稀であり，この速度低下は運用上問題となることはないと判断した[16]．

読み込み時 256 KB の部分においてファイルサイズが小さい部分において誤差が大きくなっているのは，キャッシュヒットしたデータが混在したものと考えられる．ファイルサイズが 64 KB および 128 KB の NFS による読み込みの計測値については，キャッシュヒットの影響と思われる著しく高速な値となったため，グラフからは除外した．256 KB のデータにおいても，キャッシュの影響が出たものが混在するため，計測値の分散が大きくなっているものと思われる．

6.2 GuestOS 起動時間短縮の効果

次に，電源投入から GuestOS を立ち上げ，必要なパッチ等を適用して，利用可能となるまでに必要な時間について測定した．

まず HostOS 側の起動を高速化するために，HostOS 側に対してもハイバネーションを用いることにした．電源を投入してからログイン画面が出るまでの時間は，ハイバネーションを用いなかったときには平均 111.4 秒であるのに対して，ハイバネーションの利用により平均 36.6 秒に短縮された．

続いて，GuestOS の Linux を通常の手順で起動すると，約 72 秒かかるため，これにハイバネーションを用いることにより，起動に要する時間を約 6 秒に短縮できた．同時に，4 章で述べた手順によりパッチ等を適用することにした．現在，5 種類のパッチが適用されるが，要する時間は全体で 4 秒程度，そのうち，Windows 側の共有ディスクを mount するのに要する時間が約 3 秒であった．

しかし，これらは GuestOS の起動のみを行った場合の測定値であり，実際にユーザが利用する状況においては，GuestOS の起動と同時に，Windows 側の各種プロセスの起動も同時に行われるため，ログイン処理にはさらに時間がかかることになる．

そこで実際のログインの流れに従って，電源投入の後，実際に利用可能になるまでの時間を測定したところ，GuestOS, HostOS の双方においてハイバネーションを用いなかった場合には平均 262 秒の時間が必要だったのに対し，双方でハイバネーションを用いた提案システムにおいては平均 143.3 秒に短縮された．また，「ある利用者がログオフをしたあと，別の利用者がログオンする」状況を設定して実測したところ，約 93 秒であった．

これらの測定の結果，Virtual Machine を利用したシステムにおいては，電源の投入後，最初にログオンするまでの時間は多少長いものの，電源投入をともなわない利用者にとっては十分に実用に耐える速度が実現されていることを確認した．

6.3 GuestOS に対するパッチ等の適用手順に対する評価

さらには，4.1 節で述べたパッチ等の適用手順のそれぞれに必要な処理時間を詳細に計測した．

「サーバから端末までの転送時間」を計測したところ，31.5 MB のファイルを転送するのに平均して約 9 秒の時間が必要であった．この測定値は，同時には 1 台だけのクライアントが転送した際のデータであり，複数の端末が同時に起動した場合等には，転送により長い時間がかかることになるが，4.1 節で述べたとおりユーザを待たせることはないため，大きな問題とはならない☆☆．

次の「パッチの置かれたフォルダを GuestOS からアクセスできるようにする処理」には，約 3 秒の時間が必要であった．この時間はユーザにとって待ち時間となるが，許容範囲であると考える．

最後に順次パッチを適用する処理にかかる時間であるが，5 つのパッチを順次適用するのに必要な時間は 1 秒であった．この時間もユーザにとって待ち時間となっているが，同じく許容範囲であると考える．今後，パッチの処理内容によってはさらに長い時間がかかることもありうるが，パッチの内容がユーザのログイン処理に関係しない場合には，パッチ適用処理をバックグランドで行ったり，ユーザのログインが完了してから行われるように遅延を入れたりすることが検討できる．

なお，パッチサイズが 100 MB を超えるほどに著しく大きい場合には，サーバからの転送処理に負荷が集中して問題となるものと推測される．この場合には，4.1 節に述べたとおり，HostOS を含めたディスクイメージ全体のマルチキャスト配信により対応することとなると思われるが，現在のところこのようなサイズのパッチは必要となっていない☆☆☆．

☆ 長期間の運用においてもディスクが容量不足となることがないように，現在はファイルサーバの容量の設計時の想定よりも厳しく，1 人あたり 100 MB までという容量制限をしている．

☆☆ パッチファイルの転送に際しては，再起動等により転送が途中で中断した場合においても壊れたファイルを GuestOS が利用することがないよう，ハッシュ値が適正になるまではパッチとして利用しないように配慮されている．

☆☆☆ Vine Linux に対するパッチサイズであるが，大きいサイズのものを探しても，glibc で 9.65 MB，netscape で 13 MB，emacs で 14.3 MB となっている．

7. ま と め

本論文では，Virtual Machine を利用した教育用計算機システムにおいて，Windows および UNIX の両 OS からファイルサーバに同時にアクセスがあることに着目した．この種のシステムでは，SMB，NFS の両プロトコルが混在するため一般にはファイルサーバを 1 台にまとめることは困難である．これに対して，ホームディレクトリのステート情報を利用者端末側に分散することによってファイルシステムのスケーラビリティを向上させ，大規模なシステムでも利用できるファイルシステムの構築手法を提案した．

ついで，Virtual Machine を用いて Windows と UNIX とが同時にクライアント上で稼働する大規模な教育用システムを構築する際には，OS の割当てとして UNIX を Virtual Machine 内で動作させる方が適切であることを示した．

さらに，UNIX の起動時間の短縮のためにハイバネーションを用いつつ，クライアントを最新の状態に保つために，Virtual Machine の起動後かつ利用者がログインする前に，UNIX にパッチをあてる手順を明らかにした．

さらに，これらの技術を実際に利用して，端末数約 1,200 台で構成される京都大学学術情報メディアセンターにおける教育用計算機システムを構築し，それらの技術が教育用計算機システムとして実用的な性能を有することを確認した．

Virtual Machine 技術は今も進化を続けており，今後，より柔軟性の高い製品が送り出されてくることと思われる．たとえば，文献 19) の Web サイトにおいては，VMware の GuestOS のディスクイメージを HostOS 側から直接編集するためのプログラムが開発されている．

また，Windows 上で UNIX に近い環境を実現するソフトウェアも開発されている．本文中で述べた Service for Unix は Version 3.5 の提供と同時に無償化され，今後はシステム構築の手段として利用しやすくなるものと思われる．このほか cygwin プロジェクト[20] と呼ばれるパッケージ群にも NFS クライアントが含まれている．

上記の技術の利用により，(1) Windows が NFS クライアントとして NFS サーバに直接アクセスするシステム構築技術，(2) Virtual Machine 内の OS の更新を利用者のいない時間帯に行うシステム構築技術，といった，これまでには検討外であったシステムが実用的になりつつあるように思われる．

また，本論文においては Windows を HostOS，Linux を GuestOS とする構築例を示したが，今後の Virtual Machine 技術や各 OS の実装の進化により，これらの OS の配置を逆にして，(1) UNIX 上での高速な画面描画，(2) Non Persistent な GuestOS 側で Windows を稼働させることで Windows の安定稼働，を期待するシステムにも検討の余地が出てくることも期待できる．

これらの新たに提案されている構築技術の詳細を検討し，実際にシステム構築を試みることが今後の課題として残されている．

謝辞 本システムの設計にあたって，京都大学学術情報メディアセンター藤井康雄助教授（現中部大学教授），北村俊明助教授（現広島市立大学教授）から多くのご助言をいただいたことを感謝する．また，本システムの構築においてご協力いただいた株式会社日立製作所および VMware Inc.（現 VMware and EMC Unite）に感謝する．また，本論文をまとめるにあたり，京都大学学術情報メディアセンター岡部寿男教授から貴重なご助言をいただいたことを感謝する．最後に本システムの設計をする機会を与えていただいた京都大学学術情報メディアセンター故中村順一教授に深い感謝を述べたい．

参 考 文 献

1) 丸山 伸，辻 斉，藤井康雄，中村順一：総合情報メディアセンターにおける WindowsNT による大規模分散システムの管理・運用，平成 11 年度分散システム/インターネット運用技術シンポジウム'99 論文集，pp.7–12 (Feb. 1999).
2) 吉岡 顕，田中哲朗，安東孝二：低 TCO を目指した大規模教育用システムの設計，分散システム/インターネット運用技術シンポジウム 2000 論文集，pp.1–6 (Feb. 2000).
3) 坪内伸夫：京都産業大学の情報教育と環境，私情協ジャーナル 1998，Vol.7 No.2（通巻 83 号）(1998).
4) Sugerman, J., Venkitachalam, G., Lim, B.-H., and VMware, Inc.: Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor, Proc. 2001 USENIX Annual Technical Conf. (2001).
http://www.usenix.org/publications/library/proceedings/usenix01/sugerman.html
5) 安倍広多，石橋勇人，藤川和利，松浦敏雄：VMware を用いた Linux/Windows2000 が共存する教育用計算機システムの構築，平成 13 年度情報処理教育研究集会講演論文集，pp.343–346 (Oct. 2001).

6) 安倍広多，石橋勇人，藤川和利，松浦敏雄：仮想計算機を用いた Windows/Linux を同時に利用できる教育用計算機システムとその管理コスト削減，情報処理学会論文誌，Vol.43, No.11, pp.3468–3477 (2003).
7) 竹村治雄ほか：大阪大学サイバーメディアセンター年報 2001 年度，pp.5–19 (2001).
8) VMware Workstation. http://www.vmware.com/products/desktop/ws_features.html
9) The Network Lock Manager (NLM) protocol, NLMv4, RFC1813. ftp://ftp.rfc-editor.org/in-notes/rfc1813.txt
10) Samba. http://www.samba.org/
11) Service for UNIX. http://www.microsoft.com/windows/sfu/ http://www.microsoft.com/technet/itsolutions/interop/sfu/sfu35int.mspx
12) 小田切耕司，武田保真：Samba のパフォーマンス評価およびその結果分析，ミラクル・リナックス株式会社製品本部技術部. http://www.miraclelinux.com/technet/library/lc2002_samba/index.html
13) How a Slow Link Is Detected for Processing User Profiles and Group Policy. http://support.microsoft.com/kb/227260/EN-US/
14) The Microsoft Graphical Identification and Authentication DLL (MSGINA.DLL). http://www.microsoft.com/windows2000/techinfo/administration/security/msgina.asp
15) 丸山 伸，北村俊明，藤井康雄，中村順一：5 年間使えるシステム作り，分散システム/インターネット運用技術シンポジウム 2002 論文集，pp.69–74 (Jan. 2002).
16) 丸山 伸，北村俊明，藤井康雄：Virtual Machine を活用した大規模ファイルシステム，情報処理学会研究報告 2002-DSM-25，pp.25–30 (June 2004).
17) Vine Linux. http://www.vinelinux.org/
18) IOzone. http://www.iozone.org/
19) VMware's back. http://chitchat.at.infoseek.co.jp/vmware/
20) Cygwin. http://cygwin.com/

(平成 16 年 7 月 14 日受付)
(平成 17 年 2 月 1 日採録)

丸山 伸（学生会員）

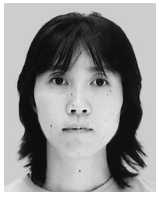
平成 10 年京都大学大学院理学研究科地球惑星科学専攻博士後期課程研究指導認定退学．平成 10 年京都大学学術情報メディアセンター教務技官，平成 11 年同助手として，教育用計算機システムの運用管理および設計に従事．平成 15 年京都大学大学院情報学研究科博士後期課程入学，同在学中．教育用計算機システムの運用技術，大規模システムの構築技術，電子メールの配送におけるセキュリティ向上技術等に興味を持つ．有限会社シー・オー・コンヴ取締役．

最田 健一

平成 15 年京都大学工学部情報学科卒業．同年京都大学大学院情報学研究科修士課程入学，同在学中．有限会社シー・オー・コンヴ取締役．

小塚 真啓

平成 11 年京都大学法学部入学，同在学中．



石橋 由子

平成元年大型計算機センター文部技官採用．平成 9 年総合情報メディアセンター異動．平成 14 年学術情報メディアセンター組織替え，現在に至る．

池田 心

平成 11 年 3 月東京大学理学部数学科卒業．同年 4 月東京工業大学大学院総合理工学研究科修士課程入学．平成 12 年 10 月同博士課程進学．平成 15 年 3 月同修了．博士（工学）．同年 4 月京都大学学術情報メディアセンター助手，現在に至る．主に最適化，意思決定システム，ゲームの研究に従事．

森　幹彦

1996 年千葉大学工学部電気電子工学科卒業．2001 年東京工業大学大学院総合理工学研究科知能システム科学専攻博士課程修了．博士（工学）．2001 年日本学術振興会未来開拓学術研究推進事業研究プロジェクトリサーチアソシエイト，東京大学先端科学技術研究センター協力研究員．2003 年京都大学助手（学術情報メディアセンター）．テキストマイニングと可視化インタフェースに興味を持つ．人工知能学会，電子情報通信学会各会員．

喜多　一

昭和 34 年生．昭和 57 年京都大学工学部卒業，昭和 62 年京都大学大学院工学研究科博士後期課程研究指導認定退学．昭和 62 年京都大学工学部助手．平成 9 年東京工業大学大学院総合理工学研究科助教授．平成 12 年大学評価・学位授与機構教授を経て平成 15 年京都大学学術情報メディアセンター教授．社会システム，ニューラルネットワーク，進化的計算，エージェントシミュレーションの研究に従事．工学博士．電気学会，電子情報通信学会，計測自動制御学会，システム制御情報学会等の会員．