CD 起動 Linux とネットワークストレージを用いた ノマディックデスクトップの性能評価

丹 英之[†] 千葉 大作[†] 上原 光晶[†] 須崎 有康[‡] 飯島 賢吾[‡]

[†]株式会社 アルファシステムズ [‡]独立行政法人 産業技術総合研究所

概要 パソコン普及率の向上，そしてインターネットのブロードバンド化によって，“ネットワークで繋がったコンピュータの遍在”が実現されつつある．この状況が実現されると，ハードウェアとネットワークは移動先で調達し，ソフトウェアはメディアを持参，ユーザのプロファイルはネットワーク越しで参照することにより実現する，放浪・遊牧型のデスクトップ環境が実用化できると考えられる．そこで，CD から起動する GNU/Linux デストリビューションである KNOPPIX と，Secure Shell を用いることでインターネット越しに利用できるファイルシステムである SHFS を組合せ，ネットワークに接続された PC であればどのマシンでも自分専用のホームディレクトリを利用できる仕組みを実装した．この実装の実用性を測る指標の一つとして，LAN 環境におけるファイルシステムの性能を測定した．本稿では，その結果について報告する．

Evaluation of nomadic desktop using CD Bootable Linux and Network storage

Hideyuki Tan[†], Daisaku Chiba[†], Mitsuaki Uehara[†], Kuniyasu Suzaki[‡], Kengo Iijima[‡]

[†] Alpha Systems, Inc.
[‡] National Institute of Advanced Industrial Science and Technology

Abstract By the spread of personal computers and broadband Internet connections, the omnipresence of computer connected to the Internet is being achieved. When such omnipresence will be realized, a nomadic desktop environment will be practical use that utilizes hardware and network existing at the destination, carries a media containing necessary software, and refers user's profile via the Internet. So, we developed a nomadic desktop environment combining KNOPPIX, which is a CD bootable Debian GNU/Linux, and SHFS, which is a network file system using Secure Shell. In this environment, any computers connected to the network can be nomadic desktop terminal accessing user's home directory. As a measure indicating practicality of this implementation, we measured performance of some file systems in LAN environment. This paper also explains this measurement evaluating the practicality of this approach.

1. はじめに

人が移動先でもデスクトップ環境を利用する手段は，大きく分けて二つある．一つは，ノートパソコンなど端末として使いたいハードウェアを常時持ち歩き，それを利用する方法である．そしてもう一つは，移動先にあるハードウェアをそのまま端末として利用する方法である．デスクトップの利用には，書類の作成や資料の参照，そしてインターネットでの検索などが挙げられる．前者の方法では荷物が増えるが，モバイル IP を用いることで移動中でも途切れることなく通信が行える．一方，後者の方法は，ユーザのネットワークへの接続性が量子的になるが，移動の際の荷物を減らすことができる．

移動先で調達した端末を自分専用のデスクトップ環境として利用するには，幾つかの方法がある．本稿では，その中の新しいアプローチである，ソフトウェアをメディアで持参し，ユーザプロファイルをネットワークで参照することで実現する，放浪・遊牧型のデスクトップ環境について述べる．

2. ノマディックデスクトップ

ノマド(遊牧民)は生活必需品を持ち歩くことにより，何処へ移動しても日常の生活を維持できる．我々は，この概念をデスクトップコンピューティングにあてはめ，ユーザが何処へ移動しても自分のデスクトップを利用できる環境をノマディックデスクトップと呼ぶことにした．このノマディックデスクトップにおけるユーザの移動とは，地理的移動だけでなく，端末間の移動も含むものと考えている．そして，ノマディックデスクトップを利用するユーザが持ち歩くものは必要最小限のものだけで，あとは移動先での現地調達で済ませる方針を立てた．すると，移動先にある端末の細かい仕様に依存することなくデスクトップ環境を実現する方法が必要になる．以下に関連する先行研究を紹介する．

2.1. ネットワークによる入出力デバイスの拡張

X Window System[1]は，サーバ・クライアントモデル(S/C モデル)で構成されている．画面表示などの入出力は，S/C 間を X プロトコルと呼ばれるネットワーク透過性

のプロトコルで通信することで行われ，分散デスクトップ環境が実現される．当時は，計算機が非常に高価であり 1 台の WS を数人のユーザで使用するスタイルが多かったことや，様々なアーキテクチャの計算機資源が混在していたため，S/C モデルによる LAN と X 端末を使った分散デスクトップコンピューティング環境が使用されていた．更に進んだ研究では，Active Office[2]がある．IR で ID を発することができる Active Badge と呼ばれるバッチを持ち歩くことで，ユーザのプレゼンス情報を取得し，最も近い端末のディスプレイにそのユーザのデスクトップ環境を再現する．

Virtual Network Computing[3]は，デスクトップ画面のキャプチャと入出力デバイスへのイベント発生割り込みを行うことでネットワーク越しのデスクトップ環境を実現する．キャプチャで得られたビットマップを RFB プロトコルで転送するので，ネットワークを流れるデータ量が多い．しかし，X プロトコルと比較すると単純であるため，クライアントが軽量である．特に，Java Applet として実装された VNC クライアント[4]は Web ブラウザ上で起動するので SSL-VPN でのリモートアクセスなどには有用である．これと同じ手法を用い，転送元となる端末の OS に最適化されたプロトコルを実装した例もあり商用化されている[5][6]．これらは，遠隔地にあるマシンのリモートメンテナンスや e-Learning でのデスクトップ管理などに用いられている．

2.2. リムーバブルメディアの運搬

上記二つの先行研究は，ユーザインタフェースである入出力デバイスの接続をネットワークで拡張したものであり，アプリケーションは全てサーバ上で実行される．それ故，マルチメディア系などの描画要求の激しいアプリケーションなどの実行には不向きである．

アプリケーションをユーザの手元にある端末で実行するアプローチとして，ソフトウェアとユーザプロファイルが入ったリムーバブルメディアを持ち歩く方法がある．これは，移動先にあるハードウェアをリムーバブルメディアから起動することで実現される．また，端末間でハードウェアとソフトウェアを共通化し，ユーザのプロファイルであるホームディレクトリのみを持ち歩く試みも検討されている[7]．そして，KNOPPIX[8]に代表される，CD-ROM 一枚で GNU/Linux が起動するライブ CD とユーザプロファイルを格納する USB のメモリーキーを持ち歩くスタイルも実用化段階にあり，製品も存在する[9]．さらに，ユーザの環境を仮想マシン上に用意することで，仮想マシンのインスタンスをそのままユーザプロファイルとしてネットワークで転送する方法も研究されている[10]．

2.3 本研究での提案

本研究で提案する，ソフトウェアの入ったメディアのみ持ち歩く方法は，ユーザプロファイルをネットワークで参照するアプローチである．今日ではパーソナルコンピュータの普及と共に，インターネットのブロードバンド化が進んでいる．これは，インターネットに接続できる PC を至る所で見られることからも明らかであろう．この状況から“ネットワークで繋がったハードウェアは遍在する”という仮定は，現実味を帯びたものとなる．すると，移動先でネットワークに接続されたハードウェアを調達し，ソフトウェアを格納したメディアを持ち歩き，自分専用のデスクトップ環境にパーソナライズするためのプロファイルはネットワーク越しで参照するアプローチが考えられる．このアプローチでは，ユーザが作成したファイルを含め，ユーザのプロファイルはネットワーク上に用意されたストレージへ永続的に保存される．そして，そのストレージを参照することでパーソナライズされた環境がユーザの居る場所で再現される．これは，ユーザに合わせてデスクトップが移動したことを意味する．そして，このアプローチはユーザプロファイル用のメディアを必要とせず，かつ，ユーザの手元にあるリソースを活用できる．

本稿の後半では，このソフトウェアが格納されたメディアだけを持ち歩く，軽く新しいコンピュータの利用スタイルの実現方法とその実用性の一評価について述べる．

3. ノマディックデスクトップの実現方法

ノマディックデスクトップは，ユーザが使う端末間を移動する自分専用のデスクトップ環境である．これをソフトウェアのみ持ち歩くスタイルで実現するには，持ち歩くソフトウェアとそれを格納するメディア，そして，ネットワークでユーザプロファイルを参照する方法について検討する必要がある．

3.1. 移動先で用いるソフトウェアとそれを格納するメディア

まず，移動先には一般的なパーソナルコンピュータである AT 互換機が在ると想定した．このアーキテクチャで動く OS からアプリケーションまでを含めたソフトウェアには，様々な種類のものがある．採用するソフトウェアは，手を加えやすく，ユーザが運搬可能な大きさのメディアに収まる容量であることが重要である．そして，そのメディアはどの端末でも読み出せることが必要となる．これらの要件から，CD メディア一枚で GNU/Linux を起動できるライブ CD の一種である KNOPPIX を用いることにした．ライブ CD であればハードディスクへのインストールも不要であるので，移動先にあるハードウェアの既存環境を破壊することなく借用できる．

KNOPPIX は，Debian GNU/Linux[11]を元に開発が進められ，日本語化[12]されたものが一般に公開されている[13]．このライブ CD はデバイスの自動認識機能に優れており，Linux でサポートしているデバイスであれば，ほぼ人の手による設定無しで利用できる．また，zlib を使った圧縮ループバックブロックデバイス cloop を用いることで 1.8GB 相当のファイルシステムが格納されており，統合デスクトップ環境 KDE やオフィススイート OpenOffice.org(OOo)など，ユーザが移動先で使用するであろうと考えられるアプリケーションは，一通り収録されている．これら，収録されているソフトウェアは全て GPL であるため再配布も可能であり，iso のイメージファ

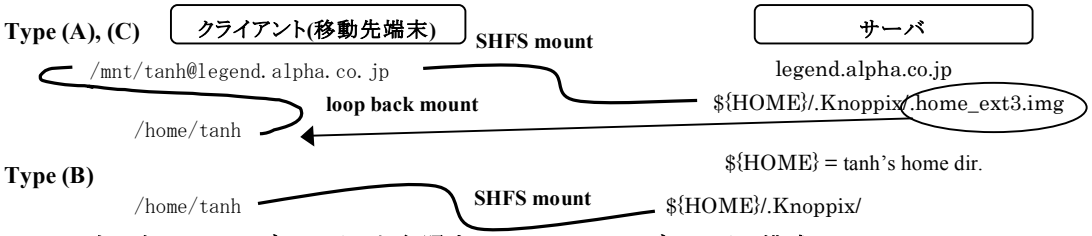

図 1. SHFS 上にあるホームディレクトリを参照する KNOPPIX のディレクトリ構成.
アカウントを"tanh@legend.alpha.co.jp"としてセッションを開始した例. 図中 Type (A),(C)SHFS 上にある Ext3FS のイメージファイル (.home_ext3.img)をループバックでホームディレクトリにマウント, Type (B)SHFS を直接ホームディレクトリにマウント.

イルがあれば直ちに複製できる. また, メディアも安価であることから, CD-ROM を破損, 紛失してもユーザの精神的苦痛は小さい. よって KNOPPIX は, 本提案のノマディックデスクトップで持ち歩くソフトウェアとメディアのセットとして, 非常に適していると言える.

3.2. ユーザプロファイルをネットワークで参照する方法

UNIX 系 OS では, ユーザが作成したファイルやアプリケーションの設定ファイルなど, ユーザ独自の環境を再現するためのプロファイルは, 全てホームディレクトリへと収納される. つまり, 何らかの方法で移動先からホームディレクトリを参照できれば, 移動先でもユーザ専用のデスクトップ環境を再現できることになる. そこで, 通信経路に SSH を使いインターネットでも安全に使用できるファイルシステムである, SHFS(secure SHell File System)[14]を用いた.

アプリケーションがネットワーク経由でファイルを参照するには, 大きく分けて二つの方法がある.

一つは, リモートにあるファイルを一端ローカルへ取り寄せ, それを参照する方法である. これには, ftp や WebDAV などを用いたカーネルレベルの実装で擬似的ファイルシステムとして扱えるものがある[15]. しかし, 使用するプロトコルはネットワークを用いた共同作業のための分散オーサリング機能に特化されており, コマンド実行時に必要となる実行アクセス権など, ファイルの細かい属性を取り扱うことができない. アプリケーションレベルでの実装では, ファイル同期のための rsync[16]がある. しかし, ローカルに無いファイルを参照するには, ファイル全体を取り寄せる必要があるので, 巨大なファイルの一部を参照する場合には不利である.

もう一つは, リモートのファイルシステムを透過的にローカルのファイルシステムとして扱うことができるようにし, それを参照する方法である. これには, 本提案で採用した SHFS の他に, NFS[17], AFS[18], SMB/CIFS[19], SFS[20]がある. また, キャッシュ制御・同期機能を拡張することでネットワークへ接続していない場合でもファイルを参照できるようにしたモバイル環境向けのファイルシステムである, Coda[21]がある. これらは, リモートにファイルサーバとしての機能を追加する必要があり, 本提案のノマディックデスクトップを利用するには事前の準備が必要となる. また, ファイルシステムの下位レイヤであるストレージデバイスを NBD[22]や iSCSI[23]を用いてネットワーク越しに扱う方法もある. しかし, ユーザ一人のホームディレクトリを扱うには, レイヤが低すぎるためオーバスペックであると考えられる.

Linux Kernel には VFS(Virtual Filesystem Switch)のレイヤがあることから, これを応用した様々なファイルシステムの実装がある. その中に, ユーザ空間のプロセスを経由する擬似的ファイルシステムが, いくつか開発されている[24]. 本提案で採用した SHFS はその一種で, ファイル操作や内容の送受信に SSH を用いるネットワーク透過型のファイルシステムである. ユーザ空間のプロセスが発行したシステムコールは, shell のコマンド文字列へ変換され, それを SSH 経由でリモート実行することによりファイルの操作や読み書きが行われる. shfs-0.33 からは, Perl で記述されたプロシージャをリモートに送信して実行することで, プロセス生成によるリモートの負荷を減少させる試みが行われている.

SHFS では, SSH でログインできるマシンをファイルサーバとして利用できる. つまり, 本提案のノマディックデスクトップは, ユーザプロファイルを置くサーバに SSH でログインできるアカウントがあれば, 既存の環境へ容易に導入できる.

3.3. ノマディックデスクトップの実装

実装は, KNOPPIX3.4 日本語版(knoppix_v3.4_20040517-20040629)に, shfs-0.31, shfs-0.33 を組込み, ホームディレクトリとして参照できるよう処理を追加した.

通常 KNOPPIX の起動シーケンスは X サーバが起動した後, ユーザ"knoppix"の権限でデスクトップ環境が立ち上がる. 今回は X サーバが起動した直後, ユーザに SSH 先となるサーバのアカウント情報の入力を促すため, 2 つのテキストボックスをもつ XDM(X Display Manager)に似たインタフェースのダイアログを表示した. ユーザはアカウント情報として, ID の部分に"ユーザ名@SSH 先のアドレス", そしてパスワードを入力する. 入力されたアカウント情報が正しいことが確認された後, そのアカウント情報を移動先の端末へ反映させる. この, サーバのユーザ認証を用いたユーザ権限の譲渡により, クライアントである移動先の端末上で, 画面ロックなど, ユーザ認証を必要とするアプリケーションを利用することが可能となる. そして, ホームディレクトリとなるファイルシステムのマウントを試み, 新たに加えられたユーザの権限でデスクトップのセッションが開始される. このように, サーバはストレージの提供元だけではなく, ユーザ権限

の認証元にもなる. そして, デスクトップのセッション終了後には, ネットワークインタフェースが無効になる前にホームディレクトリとして用いているファイルシステムのマウントを解除するようにした.

SHFS のファイルストアでは, VFS からのファイル操作に対してソケットファイルシステムを扱う処理が実装されていない. そのため, そのままホームディレクトリとしてマウントすると, KDE などのホームディレクトリへソケットファイルを作成しようとするアプリケーションでは, 起動に失敗する. そこで, ソケットファイルの位置を変えることによって SHFS を直接マウントする方法, そして, SHFS 上に置いた Linux ネィテブのファイルシステムのイメージファイルをループバックでマウントする方法を考案した[25][26]. これらの方法と shfs-0.33 から導入された Perl プロシージャによるリモート側の負荷低減の効果を測定するため下記の 3 つの方式を実装した. 図 1 はそれぞれのタイプのディレクトリ構成を示す.

- **Type A**: Shell により Ext3FS のイメージファイルを参照しループバックマウント
  shfs-0.31 を用い SSH 先で Shell を起動し SHFS 上にある Ext3FS のイメージファイル(図 1 中の.home_ext3.img)をホームディレクトリとしてループバックマウントする.
- **Type B**: Perl により SHFS を直接マウント
  shfs-0.33 を用い SSH 先で Perl プロシージャを起動し SHFS を直接ホームディレクトリとしてマウントする.
- **Type C**: Perl により Ext3FS イメージファイルを参照しループバックマウント
  shfs-0.33 を用い SSH 先で Perl プロシージャを起動し SHFS 上にある Ext3FS のイメージファイルをホームディレクトリとしてループバックマウントする.

4. 評価

実装した 3 つのタイプの中で有効なものを選別するために, LAN 環境での閉じたネットワークにおいてファイルシステムの性能を評価した. 評価では, サーバ 1 台に対しクライアントを 1 台接続し, デスクトップの起動に要する時間, そして, ファイル操作, ソースコンパイルに要する時間を測定した. また, サーバ 1 台に対しクライアントを 8 台接続し, デスクトップやアプリケーションの起動に要する時間, そして, サーバの負荷について測定した.

4.1. クライアント 1 台での評価

ホームディレクトリとするファイルシステムの違いによるデスクトップ起動までに要する時間を評価するため, クライアント 1 台で評価を行った.

4.1.1. デスクトップの起動完了に要する時間

ログインから KDE デスクトップの起動が完了するまで, つまりユーザが GUI で作業を開始することができる状態になるまでの時間, そしてクライアント側からみたデータの送受信量について測定した結果を, 表 1 に示す.

SHFS の 3 タイプの他に,

- KNOPPIX の通常起動と同じ tmpfs(仮想メモリファイルシステム)上にホームディレクトリを用意した場合
- ローカルの HD に用意した Ext3FS 上にあるホームディレクトリを用意した場合

の二つについても測定した. 通常起動で用いる tmpfs は端末の電源断で揮発するので, 起動の度に各アプリケーションの設定ファイル作成など, ホームディレクトリの構築を行う. 一方, SHFS と HD に用意した Ext3FS をホームディレクトリとした場合には, 構築したホームディレクトリは永続的なものとなる. よって, ユーザの初回起動と次回以降ではホームディレクトリの条件が異なると考えられる. そこで KDE の起動については, ユーザ初回起動と起動 2 回目についても測定した. 測定の条件は, 下記の通りである.

- 処理に要する時間は, 処理前後の/proc/uptime から処理に要する時間を求めた. 処理の終了後に, sync コマンドを発行することでカーネルにファイル入出力バッファのフラッシュを要求する. この要求が終了した時刻を操作の終了時刻とした.
- データの流量は, サーバ上で tcpdump による SSH のポートを対象としたパケットのキャプチャを行い, そのデータから求めた.

表 1. SHFS,tmpfs,Ext3FS をホームディレクトリとした場合における KDE デスクトップの起動完了までに要した時間とクライアントから見たデータ転送量.

	SHFS			tmpfs	Ext3FS
	Type A	Type B	Type C		
	time(sec)	time(sec)	time(sec)		
	TX(Kbytes)	TX(Kbytes)	TX(Kbytes)	time(sec)	time(sec)
	RX(Kbytes)	RX(Kbytes)	RX(Kbytes)		
KDE boot 1'st time	71.2	148.4	63.8	54.2	60.1
	1971.4	1767.1	1765.4		
	371.5	1917.6	359.9		
KDE boot 2'nd time	55.0	99.5	53.7	---	48.4
	1203.6	418.1	1240.2		
	950.8	634.4	922.3		

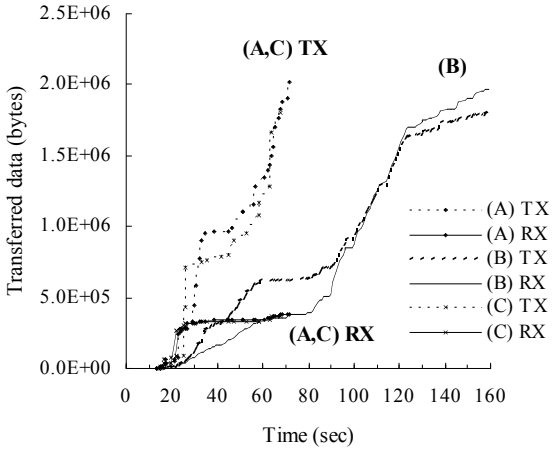

図 2. KDE ユーザ初回起動時における，サーバ-クライアント間のクライアント側からみたデータ送受信量の時間

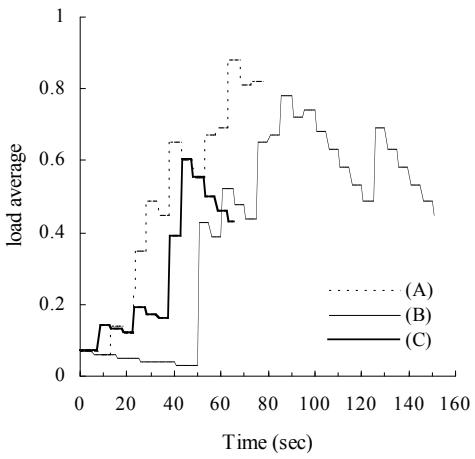

図 3. KDE ユーザ初回起動時におけるサーバの load average の時間経過.

- 機器は，クライアント側に Pentium4-2GHz, MEM:512MB, 48xCD-ROM のマシン，サーバ側に Pentium4-2.8GHz, MEM:1024MB, HD:80GB/7,200 rpm,(Linux2.6.7,XFS)のマシンを用意し，Switching HUB を経由して 100Base-TX で接続した.
- ループバックマウントする Ext3FS のイメージファイルのサイズは，100MB とした.

また，得られた結果について下記にまとめる.

- SHFS で個別にファイルを参照する場合(B)よりも，Ext3FS のイメージファイルを参照しループバックマウントする場合(A,C)の方が時間も早く，データ転送量の観点からも良い結果を得た.
- ループバックマウントする場合(A,C)は，SHFS での直接参照(B)に比べ，データの送信量と受信量に大きな差がある. これは，データを送信することで書き込んだファイルを再度読み込む際，Ext3FS 自体のファイルシステムキャッシュにヒットするため受信量が小さくなる. 一方，SHFS での直接参照(B)では，書き込んだデータはバッファリングされておらず再びサーバから取り寄せることにより，送信と受信のデータ量に大きな差が見られないと考えられる.
- KDE 起動のユーザ初回起動と起動 2 回目で比較すると，直接参照(B)では 2 回目のデータ転送量が少なかった. これは，参照するファイルのサイズよりも，Ext3FS のブロックサイズ(4KB)の方が大きく，ブロックごとによる読み書きのオーバーヘッドが生じたと考えられる.

KDE のユーザ初回起動時におけるサーバ-クライアント間のデータ転送量の時間変化を図 2，サーバの CPU 負荷を表す load average の時間変化を図 3 に示す. load average は/proc/loadaverage から採取した値で過去 1 分間の平均実行プロセス数を示す値であり，測定時瞬間の CPU 負荷とは言えない. しかし，サーバにかかる負荷の傾向を観察する上では有用な値であると考えた.

図 2 では KDE の起動完了時刻を越えてもパケットの

表 2. SHFS, tmpfs, Ext3FS 上にあるディレクトリでの各処理に要した時間とクライアント側からみたデータ転送量.

	SHFS			tmpfs	Ext3FS
	Type A	Type B	Type C		
	time(sec)	time(sec)	time(sec)		
	TX(Kbytes)	TX(Kbytes)	TX(Kbytes)	time(sec)	time(sec)
	RX(Kbytes)	RX(Kbytes)	RX(Kbytes)		
file copy [a] form tmpfs	142.8	5.1	11.3	---	3.4
	30491.2	30184.1	30491.4		
	99.9	68.8	84.4		
file extract [b] used tar cmd.	1235.8	831.7	186.7	36.9	61.4
	182709.3	170884.0	182728.3		
	35524.8	42359.5	35177.2		
kernel [c] compile	917.6	7388.5	448.6	296.8	313.1
	38841.0	55592.5	46543.4		
	103358.6	232950.7	80739.5		

[a] tmpfs に用意したサイズ 30MB のファイルを対象 FS へコピー, [b] 対象 FS にあるファイルを対象 FS 上で展開, [c] KNOPPIX デフォルトのコンフィグレーションでコンパイル(make depend && make bzImage).

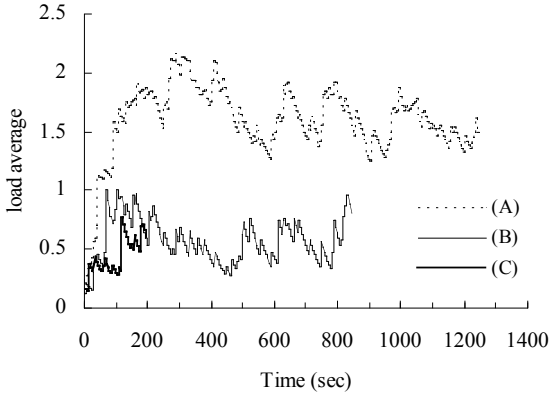

図 4. SHFS 上にあるファイル“linux-2.4.26.tar.bz2 ” (30MB)を tar コマンドでそのディレクトリに展開した場合におけるサーバの load average の時間経過.

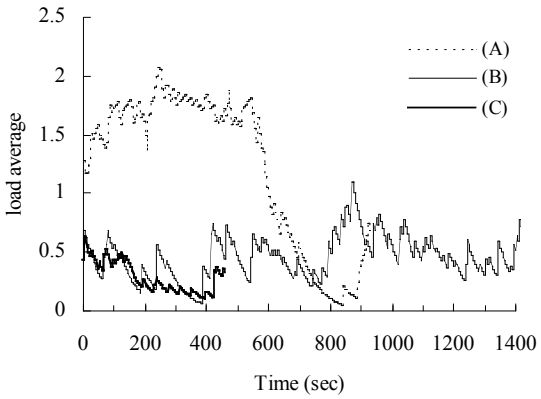

図 5. SHFS 上で linux-2.4.26 を KNOPPIX デフォルトのコンフィグレーションで make depend && make bzImage を実行した場合におけるサーバの load average の時間経

転送が見られる. これは, KDE が起動しアプリケーションのセッションを開始する時点で KDE の起動完了としたことに起因する. 直接参照(B)では 120 秒を境にクライアントの受信量が多くなっている. KDE のユーザ初回起動では, 設定ファイルを書き込む, つまりクライアントの送信量が多くなると考えられたが, 逆の結果を得た. これは, KDE 起動の初期化部分の実装に拠るものであると考えられる.

図 3 では, Shell を用いる場合(A)よりも Perl を用いる場合(B,C)の方が低負荷である. これは, Shell の場合ファイル操作や読み書きの度にプロセスを生成することによると考えられる.

4.1.2. ファイル操作, ソースコンパイルに要する時間

ファイル操作, ソースコンパイルに要する時間, そしてクライアント側からみたデータの転送量を測定した結果を, 表 2 に示す. ファイル操作には, ホームディレクトリ上へファイルをコピー, tar アーカイブの展開, そしてソースコンパイルにはカーネルのコンパイルを行った. 測定の条件は, 下記を除いてデスクトップの起動完了に要する時間の測定と同じである.

- コピー/アーカイブの展開を行ったファイルは, "linux-2.4.26.tar.bz2"(サイズ 30,772,389 bytes (30MB), bzip2 展開時サイズ 172,001,280bytes (165MB), tar 展開時 12,568 ファイル, 667 ディレクトリ)を用いた.
- ループバックマウントする場合(A,C)に用いる Ext3FS のイメージファイルのサイズを 500MB とした.
- カーネルのコンパイルは, KNOPPIX 標準コンフィグレーションでのカーネルイメージの構築(make depend && make bzImage)を行った.

また, 得られた結果について下記にまとめる.

- Perl を用い直接マウントする場合(B)では, 単一ファイルのコピーは非常に高速である. しかし, コンパイルでは Shell を用いてループバックマウントする場合(A)の約 8 倍もの時間を要する結果となった.
- 一方, Perl を用いてループバックマウントする場合(C)では, ローカルの HD にある Ext3FS を用いた場合と比較しても遜色ない時間でコンパイルを終えた. ループバックマウントする場合(A,C)は Ext3FS のキャッシュ効果により一度参照したファイルは高速に処理できるため, 直接参照の場合(B)でのコンパイル時の時間差が生じると考えられる. これは, コンパイル時におけるクライアントのデータ受信量の違いからも覗える.
- 4.1.1.における考察と同様, Shell を用いる場合(A)では読み書きの度にプロセスを生成することによるオーバーヘッドが, Perl を用いる場合(B,C)との大きな時間の違いとなると考えられる. 特に Perl で直接参照する場合(B)でのコピーは, サーバ側でファイルを開いた後, クライアントが送信するデータをそのままファイルに書き込むだけの操作で済むことからも理解できる.

ファイルを tar コマンドで展開した際におけるサーバの load average の時間変化を図 4, そしてカーネルイメージ構築の際におけるそれを図 5 に示す.
図 4,5 からも分るように, Shell を用いる場合(A)では処理の最中に 1.5 近辺の値を取り, ピーク時には 2.0 を越える. 一方, Perl を用いる場合(B,C)では, 1.0 以下で処

表 3. SHFS の各実装についてクライアント 8 台接続時における KDE デスクトップと OpenOffice.org の起動完了までに要した時間.

		Type A time(sec)	Type B time(sec)	Type C time(sec)
KDE boot	min.	123.9	152.7	57.1
	max.	150.4	217.5	94.3
	average	140.5	176.1	64.3
OOo boot	min.	96.5	22.0	14.3
	max.	106.8	36.8	19.5
	average	103.2	27.3	17.2

理が進み，しかもループバックマウントする場合(C)は負荷も少なく処理に要する時間も短い．

4.2. クライアント 8 台での評価

ホームディレクトリとするファイルシステムの違いによる，サーバ負荷の違いを評価するため，サーバ 1 台に対しクライアント 8 台を接続し評価を行った．

異なるアカウントを 8 個用意し，参照するホームディレクトリにはファイルが何も無い状態，つまりユーザが初めてホームディレクトリを利用する場合において，クライアント 8 台が同時にログインを行い，KDE のデスクトップが使用できるまでに要する時間，そして OOo の起動に要する時間を測定した．この結果を表 3 に示す．

サーバ，クライアント共に 1 台での評価と同じ条件下(内クライアント 2 台は CPU:2.8GHz を使用)で評価を行った．KDE，OOo の起動は共に，Perl を用いループバックマウントする場合(C)が早く，Shell を用いループバックマウントする場合(A)や Perl を用い直接参照する場合(B)の半分以下の時間で処理が完了する．特に，表 1 におけるローカルの HD に用意した Ext3FS パーティションをマウントして KDE を起動した場合に要する時間と比較すると，クライアントに用いたマシンのスペックの違いを考慮しても十分早く，クライアント 8 台での KDE，OOo 同時起動では，サーバ側の処理には，まだ余裕があると考えられる．Shell を用いループバックマウントする場合(A)と Perl を用い直接参照する場合(B)を比較すると，KDE と OOo の起動に要する時間が逆転する．これは，KDE と OOo の起動時にホームディレクトリをアクセスするパターンの違いに拠るものであると考えられる．

これらの測定では，処理シナリオに基づいて命令を発行し各測定を連続して行った．このときのサーバの load average の時間変化を図 6 に示す．0 秒から 200 秒の間に見られるピーク(I)は KDE の起動，そして 360 秒から 600 秒の間に見られるピーク(II)は OOo の起動に起因している．そして，660 秒で OOo の終了，720 秒からのピーク(III)は，CD-ROM に格納されている MS-Word97 形式の日本語 Word サンプルファイル(サイズ 964KB)を開いている．

Perl を用いループバックマウントする場合(C)では，ログイン直後での Perl のロードモジュール起動によりサーバの負荷が上昇するが，それ以降では殆ど負荷を生じていない．一方，Shell を用いループバックマウントする場合(A)や Perl を用い直接参照する場合(B)では，操作の度に負荷の上昇が見られる．特に，Shell を用いる場合(A)では，その上下が激しい．このことからも，Perl を用いループバックマウントする，実装(C)を用いた場合のサーバはクライアント数に対するスケーラビリティがあると言える．

サーバで Perl を用いる SHFS の実装では，プロシージャ内部でファイルアクセスの度に fopen(),fclose()を繰り返している．サーバのプロシージャで I/O 待ちが生じているようであれば，プロシージャにキャッシュを実装することを検討したい．Perl を用いループバックマウントする場合のように，クライアントが一つの Ext3FS のイメージファイルだけを参照するのであれば，fopen()後，SHFS のセッションが終了するまで mmap()を用いて参照する方法も考えられる．しかし，通常利用しているホームディレクトリをそのままノマディックデスクトップのそれとして利用する場合には，各々のファイルを直接参照しなければならない．また，万が一ノマディックデスクトップを利用できない場合のための，携帯電話や Web 端末によるホームディレクトリ上のファイルの閲覧などについて考慮すると，サーバ内部で各々のファイルを個別に保持しておくほうがよい．ファイルアクセスの速度を重視すると他アプリケーションとの連携が複雑になり，他アプリケーションとの連携を強化しようとするとクライアント側での作業に支障をきたすこととなる．つまり，一方を追求すると他方が犠牲となるのでノマディックデスクトップの適用先に合わせた実装が必要になると考えられる．

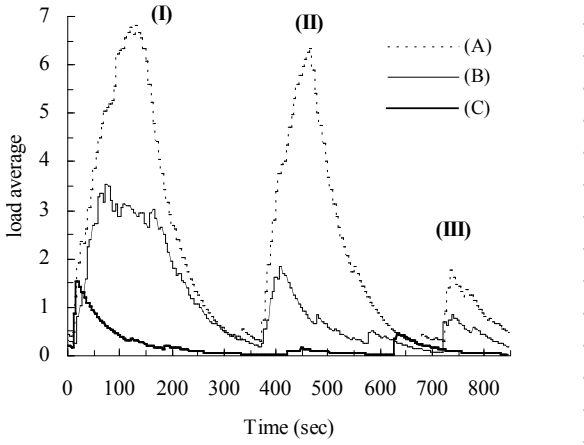

図 6. クライアントを 8 台接続した場合におけるサーバの load average の時間経過．

5. まとめ

本稿では，移動先でネットワークに接続されたハードウェアを調達し，ソフトウェアを格納したメディアを持ち歩き，自分専用のデスクトップ環境にパーソナライズするためのプロファイルはネットワーク越しで参照する放浪・遊牧型のデスクトップである，ノマディックデスクトップを提案した．そして，CD から起動する GNU/Linux デストリビューションである KNOPPIX と，Secure Shell を用いることでインターネット越しに利用できるファイルシステムである SHFS を組合せ，ネットワークに接続された PC であればどのマシンでも自分専用のホームディレクトリを利用できる仕組みを実装しその実用性について一評価を行った．この SHFS を利用したホームディレクトリの実装は，オリジナルの KNOPPIX にも取り込まれる予定である．

現在のところ，LAN 環境内における評価であり，実際の WAN 環境内での評価は行っていない．実際のインターネットでは，移動先の端末からサーバまで RTT の長い経路を通ることや，ADSL 回線での接続による非対

称リンクの経路を通ることがある．これらの通信経路では，サーバにあるホームディレクトリをファイルシステムとして直接参照するよりも，サーバにあるファイルを一端手元に取り寄せてから参照する方が適していることも考えられる．

今後は，実際の WAN 環境においてユーザのインターラクションによる評価・検証を進めていきたい．

参考文献

[1] X.Org FOUNDATION. Web Page. http://www.x.org

[2] Andy Harter, Andy Hopper, “A Distributed Location System for the Active Office”, IEEE Network, Vol. 8, No. 1, January 1994

[3] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper, “Virtual Network Computing”, IEEE Internet Computing, Vol. 2, No. 1, January/February 1998
Virtual Network Computing. Web Page. http://www.realvnc.com/

[4] ACME Labs Java page. http://www.acme.com/java/

[5] Apple Remote Desktop. Web Page. http://www.apple.com/remotedesktop/

[6] pcAnywhere. Web Page. http://www.symantec.com/pcanywhere/

[7] 木本雅彦，大野浩之，野田明生，“計算機センターは今でもユーザのホームディレクトリを預る必要があるのか?:PICKLES プロジェクトにおける携帯ファイルシステムの試み”，情報処理学会「分散システム/インターネット運用技術」研究会，May 1999

[8] Klaus Knopper, “Building a self-contained auto-configuring Linux system on an iso9660 filesystem”, the Annual Linux Showcase, October 2000

[9] LindowsCD Smile MO. Web Page. http://linspire.livedoor.com/products/mo/

[10] 須崎有康，“ネットワークを渡り歩けるコンピュータ”，日本ソフトウェア科学会，SPA2000，March 2000

[11] Debian GNU/Linux. Web Page. http://www.debian.org

[12] 須崎有康，飯島賢吾，“デスクトップとしての日本語 KNOPPIX”，情報処理学会，電子情報通信学会，FIT2003，September 2003

[13] KNOPPIX Japanese edition. KNOPPIX Japanese edition Web Page. http://unit.aist.go.jp/it/knoppix/

[14] SHFS. Project Web Page. http://shfs.sourceforge.net

[15] ftpfs. Project Web Page. http://ftpfs.sourceforge.net

[16]A. Tridgell and P. Mackeras, “The rsync algorithm”, http://samba.anu.edu.au/rsync/tech_report/tech_report.html, November 1998

[17] Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. “Design and implementation of the Sun network filesystem”, In Proceedings of the Summer 1985 USENIX, 1985

[18] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. “Scale and performance in adistributed file system”, ACM Transactions on Computer Systems, Vol. 6, No. 1, February 1988.

[19] I. Heizer, P. Leach, D. Perry, “Common Internet File System Protocol (CIFS/1.0)”, Internet-Drafts, IETF, June 1996

[20] David Mazières, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel. “Separating key management from file system security”, ACM Operating Systems Review, Vol. 34, No. 5, Decenber 1999

[21] Satyanarayanan, M., Kistler, J.J., Siegel, E.H., “Coda: A Resilient Distributed File System”, IEEE Workshop on Workstation Operating Systems, November 1987

[22] Network Block Device. Web Page. http://nbd.sourceforge.net

[23] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, E. Zeidner, “Internet Small Compuer Systems Iinterface (iSCSI)”, RFC3720, IETF, April 2004

[24] Linux Userland Filesystem. Project Web Page. http://lufs.sourceforge.net/lufs

[25] 丹英之，千葉大作，上原光晶，須崎有康，飯島賢吾，“KNOPPIX と SHFS を用いたノマディックデスクトップの提案”，情報処理学会，第 66 回全国大会，March 2004

[26] 丹英之，千葉大作，上原光晶，須崎有康，飯島賢吾，“KNOPPIX と SHFS を用いたノマディックデスクトップ環境の構築”，情報処理学会「分散システム/インターネット運用技術」研究会，March 2004