Linux Conference 2001 フルペーパ (エンタープライズ トラック)

Linux-Windows ユーザアカウント統合のための ディレクトリサービスの性能評価

Performance Evaluation of Directory Service for Linux-Windows User Accounts Integration

倉前 宏行† 島野 顕継† 木村 彰徳† 松本 政秀† 古野 良樹‡ 亀島 鉱二†

†大阪工業大学 工学部
‡シンフォニーインターナショナル

1. 緒言

近年，工学系学部においては，情報処理教育の内容が多岐にわたり講義演習における情報教育設備の実質的な使用時間も増大している．著者らはこれまで，学科独自の教育内容に柔軟に対応するための演習教室として，Windows NT と PC-UNIX のデュアルブート環境を提供する教育用 PC クラスタシステムを構築してきた [1–3].

このシステムは，コンピュータリテラシ教育からプログラミング，数値実験などの講義・演習，さらには研究利用や授業時間外のオープン利用まで，のべ 1000 名もの学生がさまざまな授業演習等で利用する．よって，ユーザはコンピュータに初めて触れる者から，研究のためのシステムソフトウェア開発を行う者まで多様であり，そのスキルやその利用形態は多岐にわたる．システムには，このようなユーザにフレキシブルに対処するような管理・運用が求められる．インターネットへの接続が普及するにつれ，フレキシビリティとセキュリティを両立させる要求も発生してきた．こうした教育用システムの運用管理については，セキュリティの確保やシステム管理の省力化を目的とした多くの研究成果が報告されている [4–9].

本システムにおいても，当初，UNIX サーバに Samba [10] を導入し，Windows のユーザ情報を NIS (Network Information Service) と連携させることにより，Windows と UNIX の 2 つの OS (Operating System) 環境においてユーザアカウントを一元化した．さらに，大学共通施設の情報センターで一括管理されている学生ユーザ情報を参照することにより，本システムではユーザアカウント管理の自動化を実現した．しかし，NIS のユーザ情報は UNIX パスワードが暗号化されているものの辞書アタックなどに対し十分な強さを持っているとは言いがたく，また Samba とのパスワード連携を行なうために構築した Wrapper もセキュリティホールとなり得る危険性を否定できない．このように，UNIX と Windows を統合してフレキシビリティとセキュリティを両立させることは容易ではない.

そこで，OS が混在したシステムにおいて，ユーザアカウントを完全に統合するとともに十分なセキュリティレベルを保つため，ディレクトリサービス NDS (Novell Directory Services) を導入した [3]．これにより情報センターが全学生に発行する計算機利用のアカウント 1 つで大学内のシステムをシームレスに利用できるようにした．しかし NDS は商用ソフトウェアソリューションであり，ソースコードなどは公開されていない．したがって，ユーザにとってはシステムの実体をつかむことはもちろん，その性能を測定することは容易ではない．そこで本稿では，本システムの管理者，すなわち NDS ユーザの立場において，ユーザアカウント情報のデータフローに基づ

いたシステムの性能測定方法を開発し，実際に測定を行なった．この結果からディレクトリサービスの性能評価について述べる．

2. OS 混在環境におけるユーザアカウントの統合管理

2.1 OS 混在環境の必要性

工学系学科においては，以下のような幅広い情報処理教育の内容に対応対応するため，UNIX 系 OS と Windows 系 OS の双方を利用する必要がある．

UNIX プログラミング教育，ネットワークコンピューティング教育

Windows 文書処理や表計算などのコンピュータリテラシ，豊富なアプリケーションソフトウェアを用いた演習・講義

これに加えて，教育用システムをインターネット接続し各種ネットワークサービスを利用するには，電子メールなどのサーバ，ネットワークの安全性を確保するためのファイヤーウォール，プロキシといった基幹サーバを，信頼性の高い UNIX 系 OS を利用して構築するのが一般的である．したがって，学生が直接操作するクライアント機はもちろんのこと，学生の目に直接触れることはないサーバ群まで含めたシステムは，必然的に OS が混在することとなる．こうした環境下でユーザを特定・認証するためのユーザアカウント (ユーザ名とパスワード) が必要となる．

本学をはじめ多くの大学では，入学時にアカウントを発行し，卒業するまで一貫してこれを使用させる．また，PC (Personal Computer) や NC (Network Computer) などの低価格化に伴い，学内共通施設の情報センターのみならず，各学科や研究室に教育用システムが自律分散し，学生ユーザは必要に応じてこれら学内のさまざまなシステムを移動する．したがって，学生は入学時に取得したアカウント 1 つで学内の全てのネットワークおよびコンピュータシステムが利用できるほうが，利便性上，望ましい．

2.2 アカウント管理の問題点

数千〜数万もの大規模ユーザ管理においては，セキュリティの確保およびアカウントデータベースの分散構築管理などといった技術的に実現しなければならない課題が存在する．

Windows 環境においては，Microsoft 独自のネットワーク認証として，NT ドメイン認証が用意されている．一方，UNIX 環境においても，NIS などがあり，ネットワークを通じた情報参照およびユーザ認証が可能である．こうした 2 つの OS 環境におけるユーザ情報 (認証用データベース) は，図 1 に示すように，それぞれの OS で独自のものであり，これを共通化し統合することは容易ではない．

NIS は暗号化されたパスワードがそのままクライアントに配布される．UNIX パスワードは暗号化されていても辞書アタックなどに対して十分な強さを持っているとは言いがたく，NIS の使用には十分な配慮が必要である [5]．また，NIS のアカウントデータベースは完全にフラットな構造をとっており，学生ユーザの入学年度や所属学科と対応させて階層的なデータ管理をすることができない．このことは，1 年に 1 度，全登録ユーザの 1/4 が入れ替わるという大学特有のユーザ管理形態においては，大変に不便である．

2.3 OS 混在環境下におけるアカウント統合方法

Windows と UNIX のユーザアカウントを統合 (一元化) しようとすると，実現しなければならない機能として

(1) Windows へログオンする際の SAM (Security Account Manager) 認証

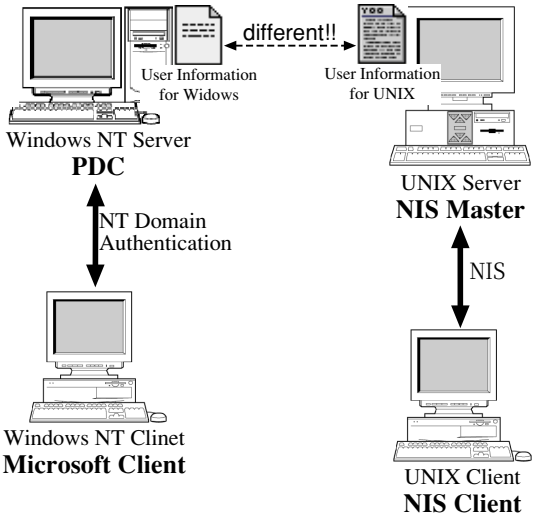


図 1: OS 混在環境におけるユーザ認証

(2) UNIX のログイン認証，および ftp, telnet, rsh, rlogin, POP などさまざまなアプリケーションにおけるユーザ認証

(3) UNIX におけるユーザ情報 (`getpwnam()`, `getpwuid()`) 提供

が挙げられる．このうち，(1) については，ドメインコントローラを用いてユーザ情報を管理するのが一般的である．ドメインコントローラは，通常，Windows NT/2000 Server 上に構築されるが，図 2 に示すように，Samba や Solaris PC Netlink [11] といったソフトウェアを導入することにより，UNIX 上に構築することも可能である．

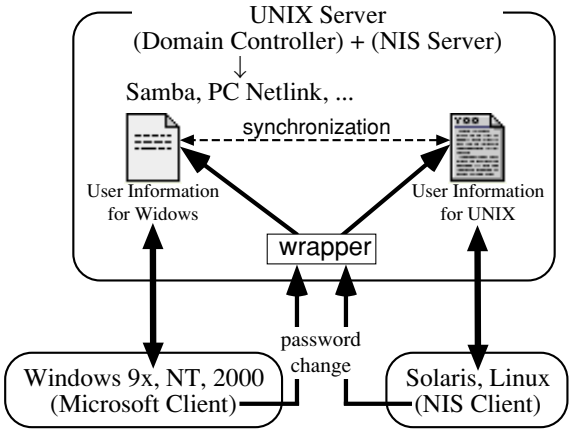


図 2: パスワード情報の同期によるアカウント一元化

したがって，同図のように，Windows と NIS の 2 つのユーザ情報の同期が図れれば，ユーザアカウントの一元化が実現される．新規ユーザの登録時に Windows および UNIX のパスワード情報を一致させておくと，後はユーザ自身によるパスワード変更が行なわれた際に，それぞれのデータベースにそれが同時に反映されれば良い．そこで，本学においては，NIS マップと Samba の SMB (Server Message Block) パスワードファイルの 2 つに対して `poppasswd` を Wrapper と

して用いたパスワード同期システムを構築することにより，OS 混在環境におけるユーザアカウントを疑似的に一元化してきた [1, 2]．同様の方法として，パスワード変更を Web ブラウザを介して行ない，CGI を用いて構築された Wrapper を経由することによりパスワード同期を図る方法もある．

しかしながら，パスワード同期のための Wrapper プログラムを悪意を持って攻撃されることや，NIS のセキュリティの甘さなど，全学規模の大規模・大量ユーザの一元管理には不安がある．

3. NDS の導入によるユーザアカウント統合

3.1 システム構成

UNIX と Windows のユーザアカウントを統合するためには，どちらかが相手の OS へ歩み寄る必要がある．たとえば，Microsoft Windows Services for UNIX [12] は，Windows ドメインコントローラを NIS サーバとして動作させることで，UNIX アカウントを Windows へ統合することができる．しかし，この方法では，パスワードとして使用できる文字種や文字数に制限があるほか，安定したシステムを運用管理する点から，Windows をサーバとして利用することに不安を感じる．逆に，Windows アカウントを UNIX 上で管理する方法としては，Samba や Solaris PC Netlink などいくつかの方法がある．しかし，システム内部が完全に閉ざされている Windows のユーザ認証方式を UNIX に合わせることは容易ではなく，アカウントデータベースが二重化され Wrapper によってパスワード同期が必要になったり，パスワード変更を一方の OS 側でしかできないといった制限が残る．

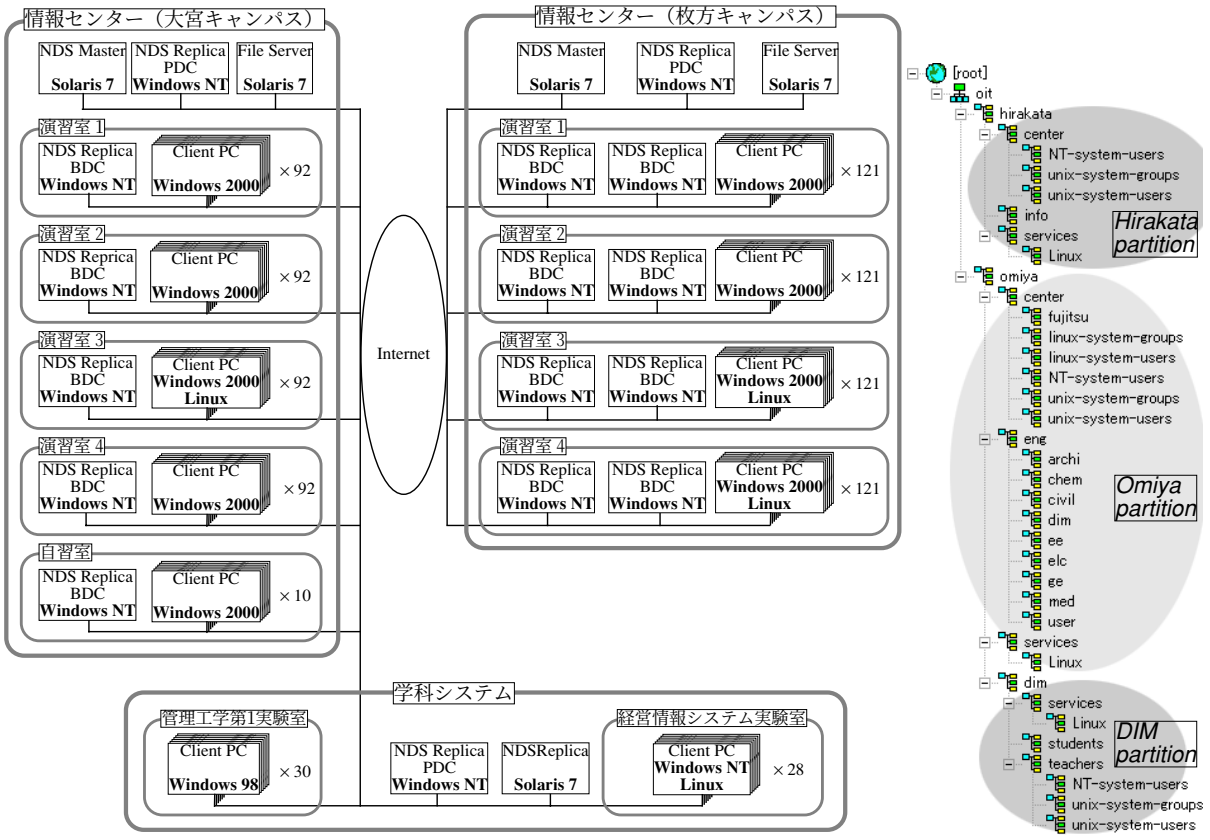
第 3 の方法は，それぞれの OS が互いに歩み寄り，2 つの OS のユーザ情報を独立して 1 つに統合するディレクトリサービスの利用が考えられる．ディレクトリサービスは LDAP (Light-weight Directory Access Protocol) などオープンな規格に基づいて実装され，OS 環境に寄らないユーザ情報管理が行なえる期待がある．そこで，全学の学生ユーザアカウントを完全に統合し，ユーザアカウントデータベースの階層管理を行なうため，図 3 に示すように，大宮 (大阪市旭区) および枚方 (大阪府枚方市) の 2 つのキャンパスの情報センターを中心に NDS Corporate Edition 8.1-1 を導入した．

クライアントは RedHat 6.2J (Linux 2.2.16) と Windows NT もしくは 2000 のデュアルブート環境を構築し，UNIX サーバは全て Solaris 7 とした．これら全てのサーバとクライアントを各キャンパスおよび学科のサブシステムの階層に合わせて，図 3 (b) のようにパーティション分割した．この階層構造に合わせて，各キャンパスの情報センターに NDS マスターサーバを置き，各パーティションごとに NT ドメイン構築のための PDC (Primary Domain Controller) を置いた．各演習室および学科のサブシステムには，それぞれ NDS レプリカサーバおよび BDC (Back-up Domain Controller) を設置した．ユーザアカウントは，教員・学生の所属学部・学科ごとに階層管理した．

3.2 Windows アカウントの NDS 統合

Windows 環境においては，Windows 9x/NT/2000 のクライアントは，ドメインコントローラ (PDC, BDC) を NDS レプリカサーバとして設定するのみで，ユーザアカウントを NDS へ統合できる．図 4 に示すように，Windows クライアントからの NT ドメインログオン要求はドメインコントローラ上で NDS へ自動的にリダイレクトされ，NDS による SAM 認証を受ける．

また，UNIX サーバに導入した Samba についても，ユーザホームディレクトリの NetBIOS (Network Basic Input Output System) マウント時などにおけるユーザ認証は，Samba を Windows クライアントとして設定し，ユーザ認証をドメインコントローラへ問い合わせることにより，NDS



(a) システム構成　　(b) パーティション分割

図 3: NDS の全学的導入

へ統合できる．

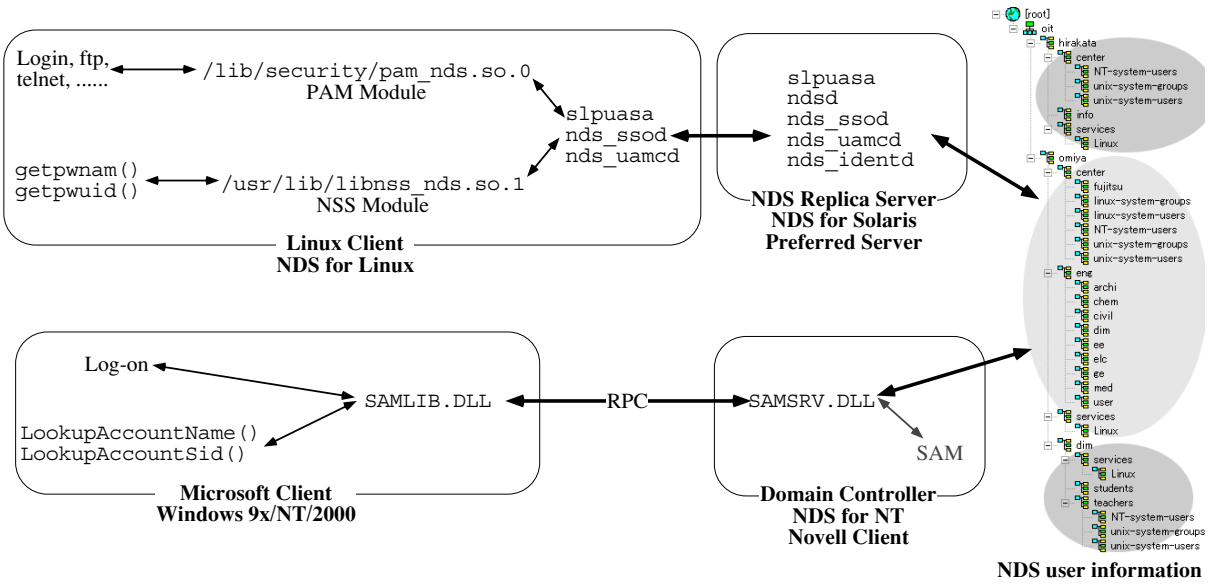


図 4: NDS を用いたユーザアカウントの統合

3.3 UNIX アカウントの NDS 統合

3.3.1 サーバの設定

NDS サーバには，NDS 共通コンポーネント，NDS NICI (Novell International Cryptographic Infrastructure)，NDS サーバ，SLP (Service Location Protocol) for NDS，NDS UAM (User Account Management)，の合計 5 つのパッケージがインストールされる．NDS のレプリカサーバにおいては，図 5 のように 5 つのデーモンプロセスが起動する．これらによって，以下の機能が提供される．

`slpuasa` SLP User Agent – Service Agent デーモン

`nds_uamcd` NDS UAM コンポーネントに用意されたデーモンで，NDS オブジェクトの FDN (Fully Distinguished Names) のキャッシュを提供

`ndsd` Novell Directory Service を提供するデーモン

`nds_identd` SSO (Single Sign-on) 操作のための NDS 識別デーモン

`nds_ssod` NDS UAM コンポーネントの SSO モジュールに用意されたデーモンで，SSO 機能を提供

```
 UID   PID  PPID  C    STIME TTY      TIME CMD
root 17060     1  0   4月 20 ?        0:39 /usr/bin/slpuasa
root 17078     1  0   4月 20 ?        1:09 /usr/sbin/nds_uamcd
root 17070     1  0   4月 20 ?       90:26 /usr/sbin/ndsd
root 17088     1  0   4月 20 ?        0:00 /usr/sbin/nds_identd
root 17091     1  0   4月 20 ?        2:01 /usr/sbin/nds_ssod
```

図 5: NDS レプリカサーバで起動している NDS 関係デーモン

3.3.2 Linux クライアントの設定

Linux クライアントにおいては，NDS 共通コンポーネント，NDS NICI，NDS サーバ，SLP for NDS，NDS UAM，の合計 4 つのパッケージをインストールする．図 6 に示すように，SLP ユーザサービスエージェント (`/usr/bin/slpuasa`)，UAM キャッシュデーモン (`/usr/sbin/nds_uamcd`)，SSO デーモン (`/usr/sbin/nds_ssod`) が起動する．このうち，UAM キャッシュデーモンと SSO デーモンは並行プロセスとして複数起動する．

```
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root       504  0.0  0.7  1440  712 ?        S    13:11   0:00 /usr/bin/slpuasa
root       516  0.0  1.1  2480 1108 ?        S    13:11   0:00 /usr/sbin/nds_uamcd
root       533  0.0  1.1  2792 1108 ?        S    13:11   0:00 /usr/sbin/nds_ssod
```

図 6: Linux クライアントで起動する NDS 関係デーモン

3.3.3 ユーザ認証

Linux や Solaris など UNIX 環境には，NDS の PAM (Pluggable Authentication Modules) 認証モジュール (`/lib/security/pam_nds.so.0` (Linux の場合), `/usr/lib/security/pam_nds.so.0` (Solaris の場合)) が提供されており，ログインをはじめ ftp, telnet, rlogin などの各アプリケーションにおけるユーザ認証を NDS へ統合できる．この設定は，`/etc/pam.conf` もしくは `/etc/pam.d/` ディレクトリ配下に置かれた PAM 設定ファイルにおいて，図 7 のように NDS PAM モジュールの指定を行なう．

```
auth      sufficient  /lib/security/pam_nds.so.0
account   sufficient  /lib/security/pam_nds.so.0
password  sufficient  /lib/security/pam_nds.so.0
session   sufficient  /lib/security/pam_nds.so.0
```

図 7: PAM 設定ファイルの記述例

3.3.4 ユーザ情報配布

NDS に登録されているユーザ情報は，NSS (Name Service Switch) フレームを通じて `/etc/passwd` ファイルの内容に相当する情報を配布することができる．これは，NDS の NSS モジュール (`/usr/lib/libndssso.so.1` (Linux の場合), `/usr/lib/nss_nds.so.1` (Solaris の場合)) が提供されている．設定は，`/etc/nsswich.conf` ファイル中に図 8 のように記述することにより，NDS の NSS モジュールを指定する．

```
passwd:     files nds
group:      files nds
```

図 8: `/etc/nsswich.conf` ファイルの設定

ただし，NDS パスワードは RSA 方式により暗号化されていることから，`getpwuid()` や `getpwnam()` などのシステムコールによってもアプリケーションには開示できないため，これらの戻り値は `*` となる．

4. システムの応答性能測定

4.1 測定の目的

前述のように，ディレクトリサービスの導入によって，OS 混在環境下においてユーザアカウント統合するとともにこれを階層的に管理することができる．本学においては，合計 900 台以上のものサーバ/クライアント機に NDS を導入し，のべ 10,000 名分のユーザアカウントを管理する．このような大規模システムにおいて実用的な性能が得られているかどうかを判断するため，さまざまな性能測定を行なった．

以下の測定は，図 3 (a) 中の学科サブシステムである「経営情報システム実験室」の環境で行なった．クライアントは富士通 FMV-6450DX3 (Pentium III 450MHz CPU, 96MB RAM, Windows NT Workstation/RedHat 6.2J OS)，UNIX サーバは富士通 GP400S5 (UltraSPARC IIi 360MHz CPU, 128MB RAM, Solaris 7 OS)，NT サーバは富士通 FMV-6550TX3 (Pentium III 550MHz

CPU, 256MB RAM, Windows NT Server OS) である．ネットワークは Fast Ethrnet スイッチ (100 Mbps) である．本実験室サブシステムは，DIM パーティションとして独立しているが，Omiya パーティション (大宮キャンパス情報センター) の読みだしレプリカとして設定している．よって，本実験室のレプリカサーバには，大宮キャンパスに所属する延べ 8,000 名分のユーザ情報を保持している．

なお，NDS の性能測定結果を比較検討するため，同実験室内に NIS マスターサーバおよび PDC の NT サーバを用意し，NDS を利用せずにユーザアカウントを NIS と PDC でそれぞれ管理した場合においても同様に性能を測定した．

4.2 Windows におけるユーザ情報参照性能

4.2.1 測定方法

Windows NT には，ユーザ情報を参照するための Win32API ライブラリ関数として

- ユーザ名から SID (security identifier) を取り出す関数

```
BOOL LookupAccountName(
   LPCTSTR lpSystemName,           // address of string for system name
   LPCTSTR lpAccountName,          // address of string for account name
   PSID sid,                       // address of security identifier
   LPDWORD cbSid,                  // address of size of security identifier
   LPTSTR ReferencedDomainName,    // address of string for referenced domain
   LODWORD cbReferencedDomainName, // address of size of domain string
   PSID_NAME_USE psUse             // address of SID-type indicator
);
```

- SID からユーザ名などを取り出す関数

```
BOOL LookupAccountSid(
   LPCTSTR lpSystemName,           // address of string for system name
   PSID sid,                       // address of security identifier
   LPCTSTR Name,                   // address of string for account name
   LPDWORD cbName,                 // address of size account string
   LPTSTR ReferencedDomainName,    // address of string for referenced domain
   LODWORD cbReferencedDomainName, // address of size of domain string
   PSID_NAME_USE psUse             // address of SID-type indicator
);
```

が用意されている．そこで，これらの関数の応答性能について測定した．

このプログラムは大きく分けて 3 つの処理からなる．

0. 測定用ユーザ名入力 検索対象とするユーザ名をあらかじめ用意しておいたファイルから読み込む．

1. LookupAccountName() 性能測定 ユーザ名を `LookupAccountName()` 関数に与え，SID を取得するのに要した時間を測定する．

2. LookupAccountSid() 性能測定 先に取得した SID を `LookupAccountSid()` 関数に与え，ユーザ名，所属ドメインを取得するのに要した時間を測定する．

時間測定は，それぞれの処理を現在時刻を取得する関数 `time()` で挟み込み，処理に要した時間 (実時間) を計測した．

4.2.2 測定結果

測定結果を表 1 に示す．ここで，比較のためドメインコントローラを NDS レプリカサーバに設定した場合と，PDC のみのサーバに設定した場合の性能を示す．

表 1: Windows 環境において 1 ユーザ分の情報を取得するのに要した時間

	NDS	PDC	性能差
`LookupAccountName()`	0.054 秒	0.024 秒	2.25
`LookupAccountSid()`	0.027 秒	0.013 秒	2.08

これより，NDS 環境は PDC の SAM データベース参照に比べ 2 倍ほど性能が低いことがわかる．

4.3 PAM 認証性能

4.3.1 測定方法

Linux 環境における NDS PAM 認証モジュールの性能を測定するため，独自の PAM アプリケーションを作成しユーザ認証に要する時間を測定した．この性能測定プログラムは，PAM ライブラリ (Linux-PAM 0.75) を用いて次のように構成されている．

1. `getpwuid()` を用いてシステムに登録されているユーザ名 (ログイン名) を取得する．
2. 1 ユーザずつ PAM 認証を行い，これに要した時間 (実時間) を計測する．

なお，ユーザ認証を行なう際，現実のユーザパスワードは不明であるので，デタラメな文字列をパスワードとして与え，認証に失敗させた．すなわち，PAM 認証のための API のうち，`pam_start()`, `pam_authenticate()`, `pam_end()` を順に呼び出しすが，このうち `pam_authenticate()` はエラーを返す．したがって，ユーザ認証に必要な `pam_acct_mgmt()` などを引続き呼び出すことはできないため，本測定は本来のユーザ認証に要する時間よりも甘い性能評価となる．

測定においては，

- まず NDS 認証を行ない，これに失敗したら UNIX 認証 (NIS による認証) を行なう設定 (NDS が推奨している設定)
- NDS 認証のみ行なう設定
- UNIX 認証のみを行なう設定

の 3 つの PAM 設定ファイルを準備し，それぞれの性能を測定した．なお，時間の測定は，測定する処理の前後に現在の時刻を取得する関数 `gettimeofday()` で挟み，処理に要した時間 (実時間) を計測した．

4.3.2 測定結果

測定用 PAM アプリケーションを実行して得られた結果を表 2 に示す．

表 2 より，NDS によるユーザ認証は UNIX 認証 (NIS による認証) に比べ 2 倍以上性能が低いことがわかった．また，NDS が想定している PAM 認証モジュールの選択順序では，まず NDS モジュールで認証を試み，これに失敗したら UNIX 認証を行なうことになっている．しかし，この方法では，UNIX 認証のみを行なうのに比べ，3 倍以上の時間を要することがわかった．

表 2: Linux 環境において PAM 認証に要した時間

	NDS+UNIX 認証	NDS 認証のみ	UNIX 認証のみ	性能差
認証回数 (認証ユーザ数)	307			
実行時間	1037.74 秒	668.03 秒	305.81 秒	
認証 1 回当たりの平均時間	3.38 秒	2.17 秒	0.99 秒	2.19

4.4 NSS 参照性能

4.4.1 測定方法

Linux 環境においてユーザ情報を参照するためのライブラリ関数 [13]

- `struct passwd *getpwuid(uid_t uid);`

  UID (User ID) からパスワードファイルの登録項目を取り出す関数

- `struct passwd *getpwnam(const char *name);`

  ユーザのログイン名からパスワードファイルの登録項目を取り出す関数

の応答性能について測定した.

このプログラムは大きく分けて 3 つの処理からなる.

1. **ユーザ全検索** UID を 0〜19,999 まで (本システムにおいてユーザとして登録されている UID の範囲) について `getpwuid()` を呼び出し，この中からユーザとして登録されている UID とログイン名を取得する

2. **getpwuid() 性能測定** 登録されているユーザの UID を連続して `getpwuid()` 関数に与え，ユーザのログイン名を取得するのに要する時間を測定する

3. **getpwnam() 性能測定** 登録されているユーザのログイン名を連続して `getpwnam()` 関数に与え，UID を取得するのに要する時間を測定する

4.4.2 測定結果

NSS 設定ファイル (`/etc/nsswitch.conf`) の `passwd` エントリを切替え，NDS を参照した場合と NIS を参照した場合のそれぞれにおいて性能を測定した．結果を表 3 に示す.

表 3: NSS 参照性能の測定結果

	NDS	NIS
ユーザ登録数	8457 ユーザ	7678 ユーザ
ユーザ全検索性能	6 時間 41 分 21 秒	12.89 秒
`getpwuid()` 性能	10 分 1 秒	5.27 秒
`getpwnam()` 性能	18 分 36 秒	5.37 秒

この結果から，1 ユーザあたりの情報を取得するのに要した平均時間を算出すると，表 4 となる．表 4 より，NDS 環境は NIS 環境に比べ 100 倍以上も性能が低いことがわかる．この性能差は,

表 4: Linux 環境において 1 ユーザ分の情報を取得するのに要した時間

	NDS	NIS	性能差
ユーザ全検索性能	1.204 秒	0.00064 秒	188,125 倍
`getpwuid()` 性能	0.071 秒	0.00069 秒	103 倍
`getpwnam()` 性能	0.132 秒	0.00070 秒	189 倍

たとえば `ls -l` コマンドを実行した際に，ファイルの所有者を表示するためファイルの i ノードに記録されている UID からユーザ名を取得するような場合に影響を受ける．

特に，NDS 環境においてはユーザ全検索性能が非常な値を示している．これは，ユーザとして登録されていない UID を `getpwuid()` に与えた際，戻り値として `NULL` が得られるまでに秒オーダの時間を要しているためである．

以上の測定結果について，NIS あるいは PDC の性能を 1 とした場合の NDS の性能比較をまとめて図 9 に示す．これより，Linux における PAM 認証と Windows における性能は，NIS あるいは PDC を用いた場合の約半分の性能しかない．さらに，NSS 参照性能は桁違いに劣悪であることがわかる．

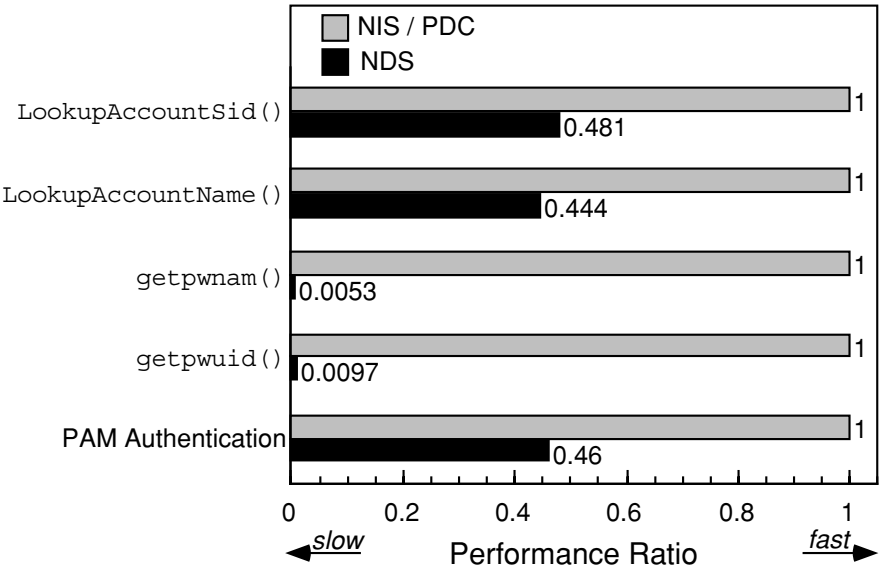


図 9: NDS の性能

4.5 負荷状態における性能測定

上述のユーザ情報参照性能の測定は，いずれも 1 台のクライアント (他のクライアントは電源 OFF 状態) から参照要求を行った際の応答性能である．したがってこれらの測定は，多人数の学生が同時に利用するという実際の講義・演習における利用形態を模擬していない．そこで，多数のクライアントから一斉にユーザ情報を参照した場合の性能を測定した．

図 10 に示すように，25 台のクライアントにおいて，`getpwuid()` 呼び出しを無限に繰り返すプログラムを実行しておき，多数のクライアントから同時にユーザ情報を参照することにより，システム全体を負荷状態とする．この状態で，応答性能測定用プログラムを利用し， `getpwuid()` および `getpwnam()` の応答性能を測定する．

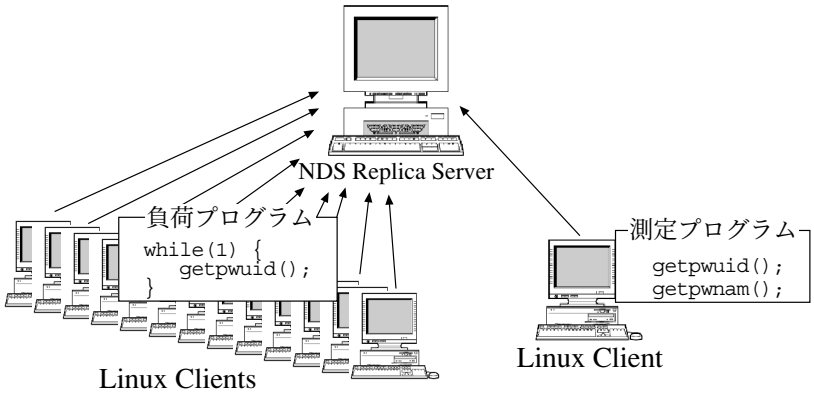


図 10: 負荷状態における応答性能の測定方法

測定の結果得られた，1 ユーザ分の情報を取得するのに要した時間を表 5 に示す.

表 5: 負荷状態において 1 ユーザ分の情報を取得するのに要した時間

	NDS	NIS	性能差
`getpwuid()` 性能	137.744 秒	0.00750 秒	18,366 倍
`getpwnam()` 性能	103.724 秒	0.00778 秒	13,332 倍

表 4 と 5 を比較すると，NDS 環境，NIS 環境のいずれにおいても，ユーザ情報の参照を同時に行うと，応答性能は低下しているが，NIS 環境においては 10 倍程度であるのに対し，NDS 環境においては数 1000 倍も低下していることがわかる．したがって，NDS 環境において多数のクライアントから同時にユーザ情報を参照する場合には，大幅な性能低下となる.

通常の Linux 環境の使用においては，ログインなどのユーザ認証の回数にくらべ，`ls -l` コマンドの実行やファイルマネージャの操作といった NSS を経由したユーザ情報の参照回数は格段に多い．したがって，NDS の NSS 参照性能の劣悪さはシステム性能の大幅な低下をまねく.

5. 結言

Linux と Windows の OS 混在環境において，セキュリティを確保するとともにユーザアカウントを完全に統合するため，ディレクトリサービスを導入した．全学規模の大規模なシステムにおいて，ディレクトリサービスの性能を評価するため，ユーザ情報を参照する際のデータフローに着目し，システム応答性能の測定技術を開発した．それぞれの OS 環境においてさまざまな性能測定を行なった結果，現時点での NDS では Linux の NSS 参照性能が桁違いに低いことが判明し，ユーザ認証のみならずシステム全体の性能が低下することが明らかとなった．今後は，NDS のバージョンアップも予定されており，他のディレクトリサービスに対しても本測定方法を適用し，大規模なユーザアカウント統合に必要なサービスシステム性能を明らかにして行く.

参考文献

[1] 倉前宏行，島野顕継，松本政秀，亀島鉱二: ネットワーク型ソフト実験のための PC クラスタシステムの設計と構築, 平成 11 年度情報処理教育研究集会講演論文集 (文部省・東北大学), pp. 139–142 (1999).

[2] Shimano, A. and Kuramae, H.: Design and Construction of Educational Computer System Using Self-maintenance System for Files and User Identification Agent, Proc. of 9th IEEE International Workshop on Robot and Human Interactive Communication, pp. 23–28 (2000).

[3] 倉前宏行，島野顕継，木村彰徳，松本政秀，亀島鉱二: ディレクトリサービスを用いた教育用 PC クラスタシステムの学生ユーザアカウント管理，情報処理学会分散システム/インターネット運用技術シンポジウム 2001 論文集, pp. 93–98 (2001).

[4] 中山仁，大西淑雅，末永正，有田五次郎: 工学系学生のための情報処理集合教育環境の設計と構築, 情報処理学会論文誌, Vol. 35, No. 11, pp. 2225–2238 (1994).

[5] 安東孝二, 吉岡顕, 田中哲朗: 大規模計算機センターのセキュリティ対策事例, 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報告, 99-DSM-16, pp. 43–47 (1999).

[6] 田中哲朗, 安東孝二, 吉岡顕: 複数 OS 環境におけるユーザ管理, 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報告, 99-DSM-16, pp. 49–54 (1999).

[7] 中山仁, 大西淑雅, 望月雅光, 山之上卓, 甲斐郷子: Linux thin client を端末とする集合教育用計算機環境の構築, 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報告, 2000-DSM-18, pp. 31–36 (2000).

[8] 斎藤明紀: 教育用大型計算機システムにおける管理の省力化手法, 情報処理学会論文誌, Vo. 41, No. 12, pp. 3198–3207 (2000).

[9] 石原進: 集合型情報処理教育施設のネットワーク設計, 教育システム情報学会誌, Vol. 17, No. 4, pp. 606–608 (2000).

[10] 例えば，Eckstein, R., C-Brown, D. and Kelly, P.: Using Samba, O'Reilly & Associates Inc. (2000).

[11] 例えば，http://www.sun.co.jp/software/interoperability/netlink/.

[12] 例えば，http://www.microsoft.com/japan/products/ntserver/sfu/.

[13] W. R. Stevens，大木 (訳)，詳解 UNIX プログラミング，ピアソン・エデュケーション，(2000), pp. 141–144.

著者紹介

倉前 宏行 1997 年九州工業大学大学院情報工学研究科情報システム専攻博士後期課程修了．同年大阪工業大学工学部助手．1999 年より大阪工業大学工学部講師．ネットワークコンピューティングによる有限要素法の大規模並列解析などの研究に従事．博士 (情報工学)．情報処理学会，日本機械学会，日本計算工学会各会員．

島野 顕継 1995 年東京水産大学大学院水産学研究科資源培養学専攻博士後期課程修了．2000 年より大阪工業大学非常勤講師．衛星通信ネットワークにおけるデータ配送プロトコルの研究開発などに従事．博士 (水産学)．情報処理学会，教育システム情報学会，コンピュータ利用教育協議会各会員．

木村 彰徳 2000 年新潟大学大学院自然科学研究科エネルギー基礎科学専攻博士後期課程単位取得退学．同年より大阪工業大学工学部嘱託講師．オブジェクト指向技術によるシミュレーションソフトウェア開発についての研究に従事．情報処理学会，日本物理学会各会員．

松本 政秀 1983 年北海道大学大学院工学研究科衛生工学専攻博士前期課程修了．同年鈴木自動車工業株式会社 (現スズキ株式会社) 入社．1995 年より大阪工業大学工学部助教授．機械構造システムの数値解析とその最適設計などの研究に従事．博士 (工学)．日本機械学会，日本経営工学会各会員．

古野 良樹 1995 年大阪工業大学大学院工学研究科電気工学専攻修士課程修了．1994 年より大阪工業大学情報センターにて嘱託職員として勤務，大学内ネットワーク及び関連サーバの構築を行う．2001 年 3 月に退職し、現在、(有) シンフォニーインターナショナル ネットワーク事業部部長．

亀島 鉱二 1973 年京都工芸繊維大学大学院工芸学研究科生産機械工学専攻修士課程修了．同年 (株) 日立製作所中央研究所入社．同機械研究所主任研究員を経て，1994 年大阪工業大学工学部教授．機械システム並びに情報システムの制御に関する研究に従事．工学博士．IEEE, SIAM, 計測自動制御学会，情報処理学会，日本機械学会各会員．