

Sambaのパフォーマンス評価およびその結果分析

小田切 耕司 武田 保真
ミラクル・リナックス株式会社
製品本部 技術部

平成 14 年 8 月 19 日

1 概要

Samba は、オーストラリアの Andrew Tridgell 氏らによって開発が行われているオープンソースソフトウェアである。Samba の機能は Windows NT/2000 が提供するサービスを UNIX で提供可能にするということであり、開発作業の開始から 10 年を経過してなお機能強化が続けられている。

もともとは、ファイル共有機能や、プリンタ共有機能などの限られたサービスを提供していたが、今日では、PDC(Primary Domain Controller) 機能や、プリンタドライバの自動ダウンロード機能など、多種多様のサービスを提供することができるようになってきている。

IA-32 マシンで動作する Linux の増加に伴って、これまで Windows によって提供されていたファイル共有サービスやプリンタ共有サービスと同等のサービスを、Linux + Samba の組み合わせで置き換えることができるようになってきた。

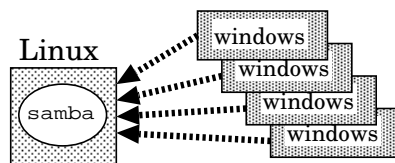
しかしながら、Windows を導入することで解決する問題に対して、Windows の代わりに Linux + Samba という組合せを導入するためには、Samba を導入したときのメリット、デメリットが明確でなければならない。特に、規模の大きな環境での導入になるほど、Windows の代わりとして新たに Samba を導入するためには、参考となる事例が無ければ、導入に踏み切りにくい。

そこで、Samba の導入事例に関して調査してみると、多数のユーザや、クライアント環境で Samba によるサービスを提供しているという情報を得ることはできるが、その導入にあたってのノウハウはなかなか得られない。

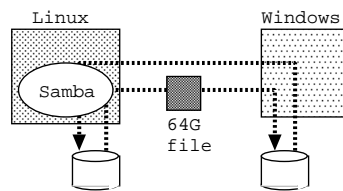
そこで、本検証では Linux と Samba の組合せに着目し、Linux 上で Samba サーバが取り扱うリソースを拡大したときに、Samba にどのような影響が発生するかを中心に検証を行った。

今回行った検証項目と、検証で想定したシステムの概要図を記載する。実際の検証作業では、Windows クライアントのかわりに Samba の提供するツールを利用した検証もある。

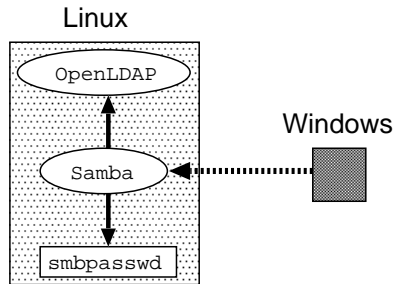
- 多数クライアントからの接続シミュレーション



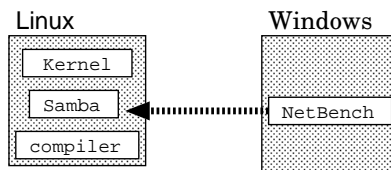
- 4 ギガバイトを越えるファイルコピー



- 多数のユーザ登録を passwd ファイルに行った場合の接続シミュレーション
- 多数のユーザ登録を OpenLDAP に行った場合の接続シミュレーション



- コンパイラを変更しての性能測定



2 多数クライアントからの接続シミュレーション

Samba サーバを構築する際の最初の問題は、どの程度のハードウェアを準備する必要があるかということである。Samba サーバは、クライアント 1 端末に対して、1 つの smbd プロセスを生成するため、クライアント数が増えるほどシステムリソースを消費する。

そこで Linux + Samba による Samba サーバ構築時のクライアント数とシステムリソースの関係について調査した。

2.1 最大同時接続可能なクライアント数

最初の検証として、Samba を利用した場合に、同時に接続可能なクライアント数の上限の調査を行った。調査は、Samba が提供する smbclient ツールを改造し、同時に複数セッションを接続することで、同時接続可能なクライアント数を調査した。

2.1.1 調査結果

Samba サーバに用いたハードウェアのスペックは次の通り。

CPU	PentiumIII 700MHz × 4
Memory	4 ギガバイト

同時接続数を増やしながら調査したところ、Samba サーバの最大同時接続可能数は 3000 であった。

この上限値について Samba のソースの調査を行ったところ、3000 という数値は、Samba のソースの MAX_SESSION_ID で定義されている数値であった。

Samba はクライアントとの接続ごとにセッション ID として一意の数値を割り当てる。このセッション ID に割り当てる値の最大値を MAX_SESSION_ID で定義している。セッション接続中は、Samba の情報を格納する TDB データベースファイルの 1 つである、sessionid.tdb ファイルに使用中のセッション ID が登録されており、新たなセッションが接続されるときには、未使用のセッション ID が割り当てられる。

その MAX_SESSION_ID が 3000 に設定されている理由を解析した結果、次の結論に達した。

Samba の utmp 機能使用時、Samba サーバにアクセスしたユーザの情報を utmp ファイルに記録するために、システムの utmp 構造体を使用している。このときに、utmp 構造体の ut_id メンバに、セッションを特定する情報として、セッション ID を格納している。

ut_id メンバは、char ut_id[4]; として定義されており、最初の 2 バイトは Samba サーバに対する接続を表すために、'S'M' の 2 バイトを割り当てている。そして残りの 2 バイトに表示可能な文字でセッション ID を識別できるようにするために、セッション ID を 2 文字に変換している。

使用可能な文字を、'0-9'a-z'A-Z' の合計 62 文字としているため、2 バイトで表すことができるセッション ID の最大値は

$$62 \times 62 = 3844$$

となる。そのため、Samba の開発者は MAX_SESSION_ID に区切りの良い 3000 という数値を選んだようである。

実際の Samba の運用を想定すると、セッション ID はクライアントの端末ごとに区別されるため、Samba サーバの限界値である 3000 セッションを同時に接続するためには、3000 台の Windows クライアントが 1 台の Samba サーバに接続するような構成となる。現実的な数値として、このような構成は考えにくいいため、同時接続可能数が 3000 という数値であっても問題無いと考えられる。

2.2 システムリソースへの影響

Samba によってサービスを提供するサーバを安定稼働させるためには、クライアントの接続によって生じるシステムリソースの消費に対応しなければならない。そこで、クライアント数の増加によって生じるシステムリソースの消費について調査した。

2.2.1 メモリ使用量

Samba サーバでは、接続するクライアントごとに smbd プロセスを fork(2) するため、同時接続するクライアント数が増えると、システムで使用されるメモリ量も増加する。

また、Windows クライアントから接続したセッションは、通常タイムアウトが発生するまで切断されないため、実際には既に使用されていないセッションでも、タイムアウトによりセッションが切断されるまでは smbd プロセスが存在するため、使用中のメモリを解放しない。

そこで、smbd がセッション接続中にどの程度のメモリを使用するか確認を行った。Samba サーバに使用したマシンは、最大同時接続可能数の調査に用いたマシンと同じである。

測定結果は、free コマンドで表示される used 欄のメモリ使用量から、ページキャッシュとして使用されている cached 欄のメモリ使用量を除いた数値を用いた。測定結果を、図 1 に表した。

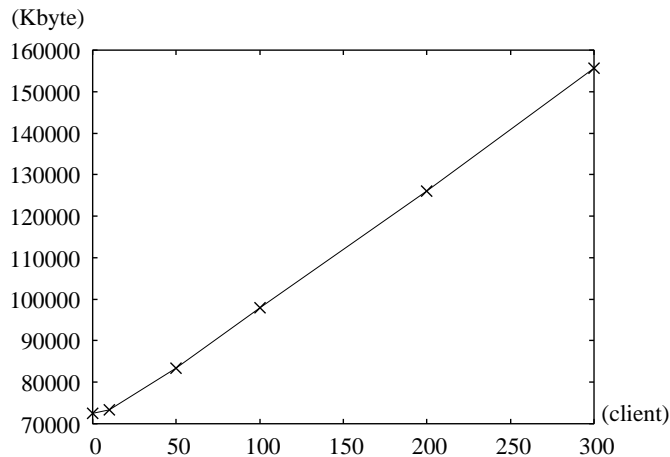


図 1: 同時接続数とメモリ使用量の関係

検証の結果より、Linux 上の Samba サーバでは、1 つの接続が確立されるごとに約 290K バイトのメモリを消費することが分かった。また、smbd、nmbd の起動時には、約 1.1M バイトのメモリを消費する。

X など不要なサービスを起動しない状態であれば、Linux は 30M バイト程度のメモリで起動する。したがって、大雑把な計算ではあるが、クライアント数とメモリ使用量の関係は、

$$\text{最低必要メモリ (M バイト)} = 30 + 1.1 + 290 * \text{クライアント数} / 1024$$

と表すことができる。

ただしこの計算では、カーネルが使用するページキャッシュのメモリを含んでいないため、実際にファイルサーバとして運用するにはメモリを増設しなければ I/O 性能に影響が生じる。

Linux カーネルは空きメモリを全てページキャッシュとして使用するため、メモリを増設するほどページキャッシュとして利用できるメモリが増え、I/O 性能の向上が期待できる。したがってファイルサーバとしての運用時には、より多くのメモリを搭載するべきである。

2.2.2 ファイルオープン数

最大同時接続可能なクライアント数の調査を実施中に、クライアント数の増加によって、Samba サーバがシステムのファイルオープン数の制限に抵触し、同時接続数が制限されるという状況に遭遇した。

そこで、クライアントが Samba に接続している時に、smbd プロセスが使用しているファイルディスクリプタ数と、ファイル転送実行中に使用されるファイルディスクリプタ数を確認した。

セッション確立後には、クライアントに対応する smbd プロセスは全部で 21 ~ 22 個のファイルディスクリプタを使用している。ファイル転送時には、ファイルディスクリプタを 1 つ使用するので、1 クライアントあたり、23 個程度のファイルディスクリプタを消費することになる。

したがって、Samba が使用するファイルディスクリプタは、

$$\text{ファイルディスクリプタ数} = 23 * \text{クライアント数}$$

として表すことができる。

Linux は、システムで使用できるファイルディスクリプタの数を `/proc/sys/fs/file-max` で確認すること

ができる。カーネルのデフォルト値は 8192 であるので、同時接続数が約 300 クライアントを越えるような環境では、この数値を拡大しておくほうが良いと考えられる。

この最大値の設定は、`/proc/sys/fs/file-max` の数値を変更することで、システム運用中に変更することが可能である。Linux カーネルの内部では、この最大値の設定は、あくまで上限値のチェックとして使用されているだけであり、上限値を拡大してもカーネルの使用メモリには影響しないので、システム起動時にあらかじめ、より大きな値に設定しておくべきである。

ただし、最初に示したように Samba の最大同時接続数は 3000 であるので、 $8192 + 3000 * 23 = 77192$ を越えるような数値を設定する必要は無いといえる。

3 4ギガバイトを越えるファイルコピー

Linux Kernel 2.4 と、Samba 2.2 の組み合わせによって、Samba サーバが取扱可能なファイルサイズの最大値が 2 ギガバイト以上に拡大された。そこで、Samba サーバ運用時に巨大なファイルのファイル転送が問題無く行われるかを確認するために、Samba サーバ、Windows クライアント間で、64 ギガバイトのファイル転送を試みた。

確認手順

1. Samba サーバ上に 64 ギガバイトのファイルを作成
2. Windows クライアントから Samba の共有フォルダにアクセスし、64 ギガバイトのファイルを Windows クライアントにコピー
3. コピーが完了したファイルを、Windows クライアントからの操作で、Windows 側から Samba の共有に再度コピー。
4. コピー前のファイルと、コピー後のファイルの MD5 値を比較

サーバ	Linux(Kernel 2.4.9) + Samba 2.2.4 日本語版
クライアント	Windows2000 Professional
ネットワーク	100Mbps イーサネット

表 1: 検証環境

結果

コピーを完了したファイルに相違は無く、問題無く転送できることを確認できた。ファイル転送に要した時間を表 2 に表す。

転送方向	転送時間	転送速度
Linux から Windows クライアントへ	144 分	7.1Mbyte/sec
Window クライアントから Linux へ	122 分	8.9Mbyte/sec

表 2: 64 ギガバイトファイルの転送時間

4 多数のユーザ環境における接続シミュレーション

従来から Samba はユーザ管理のために、`smbpasswd` と呼ばれる独自のパスワードファイルにユーザ名や、Windows 用のパスワードなどを保存していた。これはテキスト形式のフラットファイルであり、ユーザ数の増加によって、性能劣化の要因となることが指摘されていた。

これに対して、Samba 2.2 では LDAP への対応が進み、`smbpasswd` ファイルを使用せずに LDAP によるユーザ管理を選択できるようになった。

本章では、ユーザ数の増加が、従来の `smbpasswd` 形式のユーザ管理と、LDAP によるユーザ管理にもたらす影響について調査した結果を述べる。

4.1 `smbpasswd` と OpenLDAP 認証の性能比較

ユーザ数の増加が、Samba の認証にもたらす影響を調査するため、次の手順でそれぞれの認証時間の比較を行った。検証は、次の手順で行った。

1. `smbpasswd` ファイル、OpenLDAP サーバのそれぞれに、10、50、100、200、500、1000、3000、5000、10000 ユーザが登録されている状況を作成。
2. それぞれのユーザ数の状況において、ランダムにログインユーザを決定し `smbclient` を使用してユーザ認証のみ行う。
3. この認証を連続で 100 回行ったときの経過時間を計測

OpenLDAP の検索は、インデックスを作成することによって高速化が可能であるとのことであったので、Samba の認証に使用されるオブジェクトに関するエントリに関しては、インデックスの作成を行った。

また、OpenLDAP の `dbcachesize` の設定値を拡大することで、インデックスを利用した処理の性能が向上するという記述があったため、デフォルトの 100000 バイト (約 97K) を、104857600 バイト (100M) に拡大した。

以上の設定を行い、Samba サーバ、OpenLDAP サーバ、クライアントとしての `smbclient` を同一マシン上で実行して、計測を行った。

測定結果を表 3 に表す。

ユーザ数	10	50	100	200	500	1000	3000	5000	10000
<code>smbpasswd</code> (秒)	237	237	237	237	238	239	244	248	260
OpenLDAP (秒)	1002	1004	1004	1003	1002	1003	1004	1005	1006

表 3: ユーザ数と認証時間の関係

計測結果から、Samba の認証として OpenLDAP 認証を利用した時のほうが認証時間がかかることが分かる。

`smbpasswd` 形式での認証は、認証を必要としている `smbd` プロセス自身が、単に `smbpasswd` ファイルをオープンしてその内容を線形に探索していくというシンプルな処理である。一方、OpenLDAP 認証を行う場合、`smbd` プロセスは OpenLDAP のライブラリを使用して OpenLDAP のサーバプロセスである `slapd` に接続し、`slapd` の探索結果を `smbd` が受け取るというように、処理自体が複雑になっている。

OpenLDAP 認証の場合の計測結果からも、ユーザ数の増加は認証時間にほとんど影響しておらず、OpenLDAP 認証としての処理にかかるオーバーヘッドのほうが大きくなっているといえる。

計測結果より、smbpasswd 形式の計測結果では、認証時間は予想通り線形に増加しているが、OpenLDAP 認証の認証時間を越えると予想されるユーザ数は約 30 万ユーザである。

Samba を利用するようなネットワーク環境において 30 万ものユーザが存在することは考えづらい。したがって、性能の側面だけで見た場合、OpenLDAP 認証を使用するよりも、smbpasswd 形式の認証を利用するほうが有利といえる。

ただし、著者は OpenLDAP の使用方法に精通していなかったため、OpenLDAP の適切な設定やチューニングを行えなかった可能性がある。

4.2 OpenLDAP 認証のメリット

前節の結果より、性能的な面では smbpasswd 形式の認証が優れていることが検証できたが、システム運用の側面を通してみると OpenLDAP 認証の利用はメリットが多い。

Samba で LDAP 認証が本格的に利用できるようになったのは、Samba 2.2.3a からであり、Samba と LDAP 認証を利用したときの利点について述べられた情報は少ない。そこで、本節では smbpasswd 形式の認証では得られない LDAP 認証のメリットについて簡単に記述する。

OpenLDAP 認証を導入すると、Linux 上で PAM を使用して行っている認証を OpenLDAP の認証に統合することができる。

これまで、複数の Samba サーバが運用されていた場合、Samba サーバの運用形態によっては、Linux のユーザと、Samba のユーザのそれぞれをマシンごとに別々に管理する必要があった。

このような環境で LDAP サーバを導入すれば、PAM 認証と Samba の認証を簡単に統合管理することができる。

さらに OpenLDAP 認証を使用する Samba サーバを PDC として運用することで、これまで統合が難しかった Windows マシンのユーザ管理も Linux サーバ上で統合できる。図 2 参照。

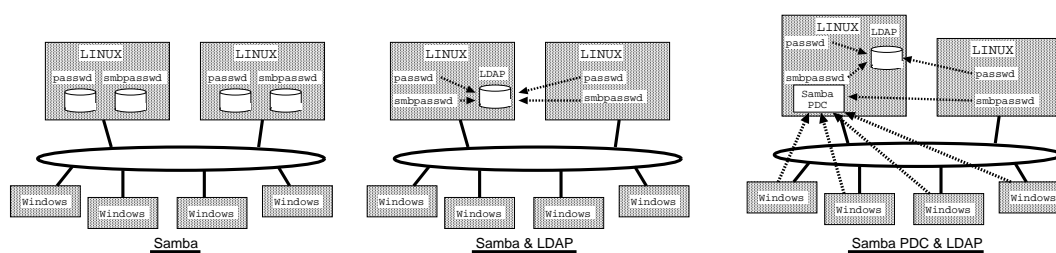


図 2: Samba と LDAP の連携

これまで Windows ネットワークに Linux サーバを導入する際の問題点として、ユーザ管理の統合という問題が大きく立ちはだかっていたが、OpenLDAP と Samba を組み合わせることで、Windows、Linux、もしくは、その他 UNIX の混在するネットワーク環境のユーザ管理を Linux で統合可能になったことは大きな利点である。

5 コンパイラを変更しての性能比較

現在、Linux 上では、GCC 2.95 系、GCC 3.1 系、Intel 製 C++コンパイラの 3 系統のコンパイラを主に使用することができる。

GCC 3.1 系や、Intel 製 C++コンパイラでは、Pentium4 向けの最適化オプションが提供されており、Pentium4 搭載マシンでは性能が向上することを期待できる。

そこで、カーネル、glibc、Samba のコンポーネントのうち、ユーザ自身でコンパイルを行う機会の多いカーネルと Samba について、再コンパイルを行い性能比較を行った。

Samba サーバのマシンには、Intel Xeon プロセッサのマシンを使用し、カーネル、Samba のコンパイルオプションで、Pentium4 向けの最適化を行った。性能の計測には、NetBench 7.0.2 を用いた。

環境

- Samba サーバ
 - Intel Xeon DP 2.20GHz x 2 HyperThreading(有効)
 - メモリ 1 ギガバイト
 - Linux kernel 2.4.9
 - Samba 2.2.4 日本語版
 - ネットワーク 100Mbps イーサネット

使用したコンパイラとそのオプションの組み合わせを表 4 に表す。カーネルについては Intel 製コンパイラではコンパイルできないため、組合せに含んでいない。

カーネル	Samba
GCC 2.95.3 -march=i686	GCC 2.95.3 -march=i686
GCC 2.95.3 -march=i686	GCC 3.1.1 -march=pentium4
GCC 2.95.3 -march=i686	Intel Compiler -tpp7
GCC 3.1.1 -march=pentium4	GCC 2.95.3 -march=i686
GCC 3.1.1 -march=pentium4	GCC 3.1.1 -march=pentium4
GCC 3.1.1 -march=pentium4	Intel Compiler -tpp7

表 4: コンパイラとオプションの組み合わせ

測定は、1 台のクライアントに対する転送性能の最高値を計測した。したがって、Samba サーバ側では、1 つの smbd に対して、絶え間無くリクエストが要求される状態となる。

測定の結果を図 3 に表す。

測定結果から、コンパイルオプションの違いによる性能向上を確認することはできなかった。

そこで、ベンチマーク実行中の smbd プロセスのプロファイル情報を取得した。

表 5 に smbd プロセスのプロファイリングの結果から、実行時間の多い関数を順に並べた。

実行時間の多い関数の上位の内訳をさらに詳しく調査したところ、_IO_default_xsputn、_errno_location、memcpy、vfprintf の処理の呼び出しのほとんどがログに記録するための書式の解析、ログ文字列の作成であった。そこで Samba の設定を確認したところ、ログレベルを 5 と設定していたため、ログの処理に時間がかかっていたことが判明した。

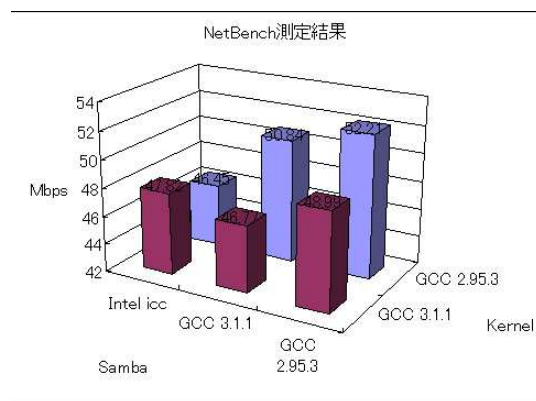


図 3: コンパイラの変更による性能への影響

関数名	呼び出し回数	1回あたりの実行時間
_IO_default_xsputn	70290739	0.00
__errno_location	46108739	0.00
memcpy	18416181	0.00
vfprintf	17189802	0.00
funlockfile	14130247	0.00
flockfile	14128673	0.00

表 5: smbd プロセスのプロファイリング結果

改めて、ログレベルを 0 にして再測定をしてみたところ Samba の転送性能が明らかに向上した。表 6 参照。

ログレベル	転送速度	CPU 使用率
5	52.2Mbps	24%
0	69.4Mbps	8%

表 6: ログレベルの設定変更による性能測定結果

本検証の目的であるコンパイラの最適化による性能比較に関しては、十分な検証が行えず、有意な結果を得ることができなかった。しかし、その検証の過程において、Samba 運用時のログレベルの設定が、システムの CPU 使用率と転送性能に対して大きな影響を及ぼすことを改めて確認することができた。

6 まとめ

本検証の結果、Windows サーバを Linux + Samba を置き換えるにあたって、Samba やカーネルのソースの制限が問題となる可能性は低いことが確認できた。

また、OpenLDAP 認証を使用することで、Linux と Windows の混在するネットワーク環境において、Linux 中心のユーザ管理が可能であることは、Linux の更なる普及に大きなインパクトをもたらすものと考えられる。

しかしながら、Samba と LDAP の連携機能に関しては有用な情報も少なく構築作業はなかなか難しいものであった。OpenLDAP を利用した認証時間の計測では、適切な設定を行うことができていなかった可能性もあり、Samba と OpenLDAP の連携に関して継続して調査する必要があるといえる。

同様にコンパイラの変更による性能比較は、最終的な結論を下せるほどの解析が行えなかったため今後の調査課題である。

7 謝辞

検証環境をご提供いただいた OSDL、ならびに施設の利用に当たって多大なご協力をいただいた OSDL のスタッフの方々に感謝いたします。Samba 日本語版の開発にあたってご指導、ご協力いただいている全ての方々に感謝いたします。そして Samba をはじめとするオープンソースソフトウェアの開発に携わる全ての人々に感謝いたします。

8 参考文献

1. Samba-PDC LDAP v.3 howto
http://www.unav.es/cti/ldap-smb/ldap-smb-2_2-howto.html
2. OpenLDAP 2.1 管理者ガイド
<http://www.interq.or.jp/earth/inachi/openldap/admin/intro-ja.html>
3. Linux*版 インテル (R) C++ コンパイラ
<http://www.intel.co.jp/jp/developer/software/products/compilers/c60l/index.htm>
4. 高橋基信: アンドキュメンテッド Microsoft ネットワーク, 翔泳社 (2002)
5. 鵜飼文敏: 大特集 認証は LDAP で, UNIX USER(2002/1), ソフトバンクパブリッシング