規模に応じたインターネットサーバー構築・運用ノウハウ

民田 雅人 （松井証券株式会社）

1998年12月16日

InternetWeek 98　**国立京都国際会館**

(社)日本ネットワークインフォメーションセンター編

目的

世の中には大規模なものから小規模のものまで様々なインターネットサーバーがあります。インターネットサーバー構築・運用のノウハウはなかなか身につきにくいもので、実際に失敗してみないと駄目です。私も過去何回となく失敗しています。 その体験を踏まえて、実例を挙げながら説明したいと思います。本チュートリアルが、適正なシステムを適正なコストで運用する為のヒントになれば幸いだと思っています。

目次

1.はじめに

今回は、最近手に入りやすく高速になったAT互換機と、Sunのワークステーション（Solaris2.x)を使用したインターネットサーバー例を中心に説明します。

規模とコスト

概して小規模なシステムは構成も安上がりになりますし、手間もあまりかかりません。しかし、大規模なシステムはシステムを組むのに費用もかかりますし、運用にも多少手間がかかります。ノウハウが必要になります。もちろん大きなシステムを小さなコストで運用できれば理想的ですが、現実はなかなかそうはいかないようです。

早い事は良い事だ

CPUは速い方が計算が短時間で終わります。ハードディスクが速ければデータの読み書きが短時間になりますし、スワップのパフォーマンスが上がりアプリケーションの起動も速くなります。ネットワークが速ければデータ転送が短時間で終わります。速ければ多くの場合、マイナスになることはないと思います。

スピードとコスト

速いシステムを作るというのは大規模なシステムになることが多いので、概してイニシャルコストが高くなり、遅いシステムは安く済みます。しかし、もう一つ重要なのはランニングコストです。速いシステムをつくっておくと手間はかかりませんから、管理者の負担が軽くなりランニングコストが安くて済みます。 イニシャルコストを押さえるために安いシステムを作って喜んでいても、すぐに限界がきて次の手間をかけるようですと、ランニングコストが高くついてしまうことになります。ですから、この辺をきちんと見きわめて作成する必要があります。

インターネットサーバーの例

サーバーにも様々な種類がありますが、今回はインターネットサーバーの例です。
インターネットサーバーにも、WWWサーバー、ネームサーバー、メールサーバー、NEWSサーバー、WEBのキャッシュサーバー、IRCサーバー等がありますが、今回はWWWサーバーとメールサーバーとNEWSサーバーを中心に説明します。
ネームサーバーは、大規模なシステム（1,000ドメイン位セカンダリを受け持つ一部のISP）は別として、一般には重くないですですから、比較的簡単に運用できます。
IRCサーバーはそれなりに面白いのですが、個人的には速いマシンを持ってくれば終わりだと思っていますので、今回は触れません。

２．システム構成

インターネットサーバーを組む場合のシステム構成、パーツの選び方について説明します。

システム構成のポイント

普通のインターネットサーバーシステムはCPU、メモリ、ハードディスク、ネットワーク、等をまとめ上げたパッケージになっています。 これらの構成要素選択が重要なポイントになります。

・ＣＰＵの種類とスピード
・メモリの容量
・ハードディスクの台数と容量
・Network I/Fのスピード(10Mbpsか100Mbps)と置く場所

ＣＰＵ選択のポイント

CPUを選択する場合の選択ポイントは次の通りです。

・アーキテクチャ
・クロック
・キャッシュのスピードと容量
・メモリ実装容量

ＣＰＩアーキテクチャ

主なＣＰＵアーキテクチャーとメーカー名を示します。

・x86、i386　PC-AT互換機
・SPARC　Sunとその互換機
・PowerPC　Macintosh　インターネットサーバーとしては不向き
・MIPS　ＳＧＩ（シリコングラフィックス）
・Alpha　COMPAQ
・PA-RISC　ＨＰと日立

AT互換機のCPU

AT互換機、i386アーキテクチャのCPUクロックを示します。

・486　25～100MHz
・Pentium　75～200MHz
・Pentium MMX　166～266MHz
・Pentium-Pro　150～200MHz
・Pentium-Ⅱ　233～450MHz
・Pentium-ⅡXeon　400～450MHz

CPUベンチマークで使われているものにSPECintがあります。 SPECint95とは95年のSPECintベンチマークテストでの数字です。 この数値で見ますと、一番速いCPUと一番遅いＣＰＵでは0.3～15の開きがあり、50倍ぐらいの能力差があることになります。

SPARC

SPARCの種類とクロックを示します。

・ MicroSPARC(LX,Classic)　　50MHz
・ MicroSPARC-Ⅱ　　70-110MHz(SPARC Station　5に搭載)
・ TurboSPARC　　170MHz
・ UltraSPARC　　167MHz
・ UltraSPARC1+　　200MHz　　(現行のSPARCアーキテクチャ)
・ UltraSPARC Ⅱ　　250～333MHｚ　(現行のSPARCアーキテクチャ)

UltraSPARC ⅡIもあります。 又、600MHzのＣＰＵもアナウンスされていますが、まだ製品には組み込まれていません。 単純にクロック比で見ても６倍以上ありますが、実際のCPU能力差はもう少しあると思います。

CPU Clock Speed

CPUクロックのスピードは単純な指標だと思って構いません。当然クロック数が多いもののほうがＣＰＵは速いのですが、アーキテクチャの違いがありますと、クロック数が２倍になっても速度は倍以上になることもありますし、ならないこともあります。
例えば100MHzのPentiumと200MHzのPentiumを比べると、倍程には差はありません。それは単純なCPUクロックの比だけでなくて、外部バスの違いも絡んでくるからです。100MHzのPentiumと200MHzのPentium-Proを比べると、多くの場合Pentium-Pro200MHzの方が倍以上の速さです。

CPU能力の違い

私が過去に測定した記憶を記述したもので正確な数字ではないのですが、FreeBSDカーネル(GENERICではなくて適当に削ったカーネル)をコンパイルする時間です。

・Pentium-Ⅱ/333　　２分程度
・Pentium-Pro/150　　４分から５分程度
・Pentium/120　　10分程度
・486DX2/66　　20～40分程度

メモリ容量やハードディスクの種類等の構成が異なるので一概に比較できませんが、この程度の目安だと思ってください。

Cache Memory

CPUのアーキテクチャと密接な関係があるのは、キャッシュです。 CPUはキャシュまで含めて設計されています。 キャッシュはCPUとメモリのバス速度の差を吸収する役割を果たしています。
CPUのほうがメモリより圧倒的に速いので、キャッシュがバスの速度の差をうまく吸収しないとCPUは速く動きません。
キャッシュには１次キャッシュと２次キャッシュがあり、１次キャッシュはCPU内蔵で容量は4～64Kbyte位、２次キャッシュの多くはCPU外部にあり、容量は128Kbyte～２Mbyte位です。多くはと書いたのは、内蔵されているCPUもいくつかあるからで、例えばPentium-Proは完全にチップの中に内蔵されています。SPARC系だと内蔵されているものはないと思うのですが、CPUのすぐそばにモジュールとしてくっついています。
キャッシュの能力を判断するのは、まずキャッシュサイズ、つまりメモリ容量です。もう一つ重要なのがキャッシュバススピードで、キャッシュがCPUとメモリのスピード差を吸収する役割を果たすので重要になります。もう一つ見落としてならないのがキャッシュ可能エリアです。 この大きさはCPUによって異なってきます。

キャッシュサイズ(x86)

CPU毎のキャッシュサイズの例です。

・Pentium-Pro １次キャッシュ（プログラムコード用８KB、データ用８KB）合計16KB

２次キャッシュが256KB～１MB

・Pentium-Ⅱ ～300MHｚのCPU １次キャッシュ（コード用16KB、データ用16KB）

２次キャッシュが512KB

300MHｚ以上のＣＰＵ １次キャシュ（コード用16KB、データ用16KB）

２次キャッシュが１MB

AMDのK6のデータも説明します。

・ＡＭＤ K6 １次キャッシュ(プログラムコード32KB、データ用32KB)

２次キャッシュはCPUとは別のマザーボード上に構成しますので、マザーボードを何を選ぶかによってキャッシュの量は決まってきます。

キャッシュサイズ(SPARC)

SPARCの場合を次に示します。

・UltraSPARC-ⅡI（Ultra5やUltra10で使われている）

１次キャッシュ （コード用16KB,データ用16KB）

２次キャッシュ 270MHz 256KB,300MHz 512KB,333MHz 2MB

２次キャッシュの容量が多いほど、CPUとしてはトータルのスループットが上がりますので、270MHｚのものと333MHｚのものを比べた場合、このクロック差以上の性能差があらわれるはずです。

・UltraSPARC-Ⅱ（ Sunの大型のマシンで使われている）

１次キャッシュ （コード用16KB、データ用16KB）

２次キャッシュ 250MHｚ - １Mbyte、300MHｚ-２Mbyte

UltraSPARC-Ⅱiより一回りキャッシュ量が多く設定されているので、同じクロックで比べた場合、パフォーマンスは少し上がるはずです。 SPECint値で１まで変わりはしないのですが、11だったものが12になっている位の差があったような記憶があります。

Pentium-Ⅱは速いか？

「Pentium-Ⅱは速いか？」という話があります。Pentium-Ⅱのクロックスピードは233MHｚから333MHｚまで、これは外部バスクロックが66MHｚのものです。今は外部バスクロックが100MHｚになっているものが出ていますので、350MHｚや400MHｚのクロックが得られます。Pentium-Proは200MHｚまでしかありません。重要なのはPentium-Ⅱの333MHｚあるいは400MHｚまでのXeonになっていないアーキテクチャのＣＰＵは２次キャッシュのクロックスピードがCPUクロックの半分で動きます。

Pentium-ProですとキャッシュクロックはCPUクロックと同じスピードで動きます。したがって、Pentium-Proの200MHｚはキャッシュクロックが200MHｚで動いており、そのスピードでデータの読み込みが可能なのですが、Pentium-Ⅱですと、300MHｚのＣＰＵを持ってきたところで、２次キャッシュのスピードは150MHｚのクロックに同期して動くことになり、キャッシュにヒットし続けている場合は外部メモリとやり取りするときのスピードは変わらないのですが、ヒットせずに２次キャッシュから読み込む部分についてはP6-200のほうが速くなります。

計算するだけならPentium-Ⅱのほうが速いのですが、大規模なデータを扱うようなアプ

リケーションになるとPentium-Proのほうが速くなるケースが多くあります。といっても、これは実際に載せるメモリの量も絡んできて単純ではないのですが、Pentium-Ⅱの266MHｚ位ですとPentium-Proのほうが速い場合が多いと思われます。この件についてはレポートがいくつか出ております。

メモリバス
メモリバスと書いてありますが、CPUとメモリのインターフェースの事で、メモリバスのクロックとバス幅で転送速度が決まります。バスの幅が32bitでクロックが33MHｚですと、理論的に転送できた場合には132Mbyte/sec、つまり１秒間に132Mbyteのデータを転送することができます。これが128bitでクロックが100MHｚの場合は1,600Mbyte/secのデータが転送できますので、単純に見ると10倍ぐらい違います。

実際のメモリの挙動
単純に見ると10倍ですが、実際にそれだけ違うかどうかはわかりません。 メモリは必ずしもクロックに同期して１ワードずつ転送できるわけではないからです。 最初のアクセスに少し時間がかかり、そのあと連続して３ワードぐらいまでは速いのですが、その次に時間がかかるのです。 PCを触った経験のある人ですとマザーボード設定に、4-2-2-2とか5-1-1-1というのを見かけた事があると思います。 つまり4-2-2-2のケースですと、４ワード転送するのに、最初の１ワードに４クロック、残りの３ワードは全部で６クロック、合計は10クロックです。5-1-1-1のケースでは、最初の１ワードに５クロックかかり、残りの３ワードの転送には１クロックずつの３クロックで、トータルで８クロックかかります。ですから、連続してアクセスする場合には後者のほうが少し速いのです。でも、ランダムにアクセスするとこのようなメモリサイクルでは動かないケースがあります。 １ワードずつランダムに読むケースでは、最初の１ワードは必ず４クロックかかり、２番目は５クロックで１個遅いので、この場合ですと１番目のほうが速くなります。これはアプリケーションに異なるので一概には言えません。

メモリ容量
多くの場合はメモリスピードよりもメモリ容量のほうが問題になります。メモリというのは必要にして十分なメモリを用意するというのが基本です。多過ぎるメモリというのは単なるお金のむだ遣いになるのですが、少ないとこれはパフォーマンスに影響します。
メモリ容量選択のポイントですが、稼働するプロセスの使うメモリ、OSの利用するメモリ、ディスクバッファ、この辺を見きわめてメモリの実装量を計算する必要があります。例えば１つのプロセスが１Mbyteのメモリを使い、プロセスが100個動くような場合ですと、最低100Mbyteはプロセス用にメモリを確保しなければなりません。OS用のメモリもちょっと確保する必要があります。OSの種類によって容量は異なって来るのですが、カーネルとそのカーネルのワークエリアは10Mあるいはもっと少なくていいはずで、１Mとか２M位だと思います。 そして、残りの部分がディスクバッファにるので、これをどの位見越すかでメモリ容量が決まります。トータル量については実際にやってみないと何とも言えないところがあります。

ハードディスク
CPU、メモリときまして、次はハードディスクです。ハードディスクはご存じのようにヘッドが磁気円盤上の磁気情報を読み出したり書き込みを行ったりするものです。磁気円盤ですから円周方向には読み取るだけでして、円盤の外周部と内周部に対してはヘッドがシークという動作を行います。シークしている間は全くデータが読めないわけですから、連続して読み出すのは割と速いのですが、ランダムにアクセスしますと、ヘッドが物理的に

動く時間が必要ですので、桁違いに遅くなります。ランダムアクセスはインターネットサーバーのボトルネックの要因になりますので、ランダムにヘッドを動かさないような工夫をする必要があります。それを何も考えないと、サーバーとしては遅くなるという事です。インターネットサーバーには一般的にはSCSIというインターフェースが使われています。何故SCSIが使われるかというと、スピードが速くて、１つのCPUに多くのハードディスクが接続できるからです。PCでＩＤＥが使われていますが、ＩＤＥはインターフェース信号の規格上大量にハードディスクを増設するのには向きません。

実際のＨＤＤの例

実際にIBMのDDRSシリーズ（34560/4.3GBと39130/9GB）のカタログスペックを示します。

a. 109-171 Mbit/sec Media data rate
b. Rotational speed 7200rpm
c. Sustained data rate 8.3-13.3 MB/s
d. Average seek time 7.5ms
e. Average latency 4.33ms

皆さんあまり詳細には読まないと思われるのですが、a.は、ハードディスク上の円盤上の内周部や外周部そのままの回転数と密度から計算されるメディアの読み出す量です。この値は生のレートですから、フォーマットしたあとにセクタ等をつけ加えると、実際にはここまでのスピードは出なくなります。スペックを見て判りますようにＨＤＤはメモリに比べて相当スピードが遅いと言わざるを得ません。

SCSIコントローラ

ハードディスクを接続するのがSCSIコントローラです。 SCSIコントローラはDMA(Direct Memory Access)です。AT互換機の世界ではバスマスタと呼ばれますが、DMAはCPUを介さずにデータ転送を可能にします。プログラム（CPU）がメモリ上にデータを用意し、コントローラに対してコマンドを送ると、あとはコントローラがメモリとディスク間のデータ転送を行ってくれます。コントローラがデータ転送を実行している間は、プログラム（CPU）は次の作業をする事ができるので、SCSIはプログラム(CPU)動作の妨げにはなりません。しかし、注意しなければならない点があります。 ハードディスクに対して書き込みを行う(WRITE)場合は、コマンドをコントローラに送れば、書き込み動作の完了を待つ必要はないのですが、読み出し（ＲＥＡＤ）の場合は、読み込んだ結果（データ）が欲しいわけですから、どうしても動作完了を待たなければなりません。つまり、読み出しコマンドはパフォーマンスには影響を与えるので注意が必要となります。２台のSCSIコントローラに２台のハードディスクを接続した場合には、１台のSCSIコントローラに２台のハードディスクを接続した場合よりも、書き込み速度は倍にすることができます。つまり、一つ目のコントローラに書き込みのコマンドを送って、すぐ２つ目のコントローラにも書き込みのコマンドが送れますので、倍近い性能を得る事ができます。ところが、読み出しは結果を待っていなければならないので、コントローラを複数にしてもあまりメリットがありません。

Network I/F

現在サーバーをネットワーク（Etherねｔ）につなぐ場合、ほとんど選択の余地がありません。せいぜい考えることは10Mにするのか100Mにするのかという事位です。 現在は10baseTや100baseTXを使うケースがほとんどなので、良いEther Switchと組み合わせてfull-duplexにすれば、パフォーマンスは少し向上します。良いSwitchと書きましたけれど

も、full-duplexのできない、あまりよくないSwitchもあるので注意してください。その他、100Mのインターフェースで FDDIがありますが、どうしてもコストが非常に高くなりますので最近はあまり使われていません。

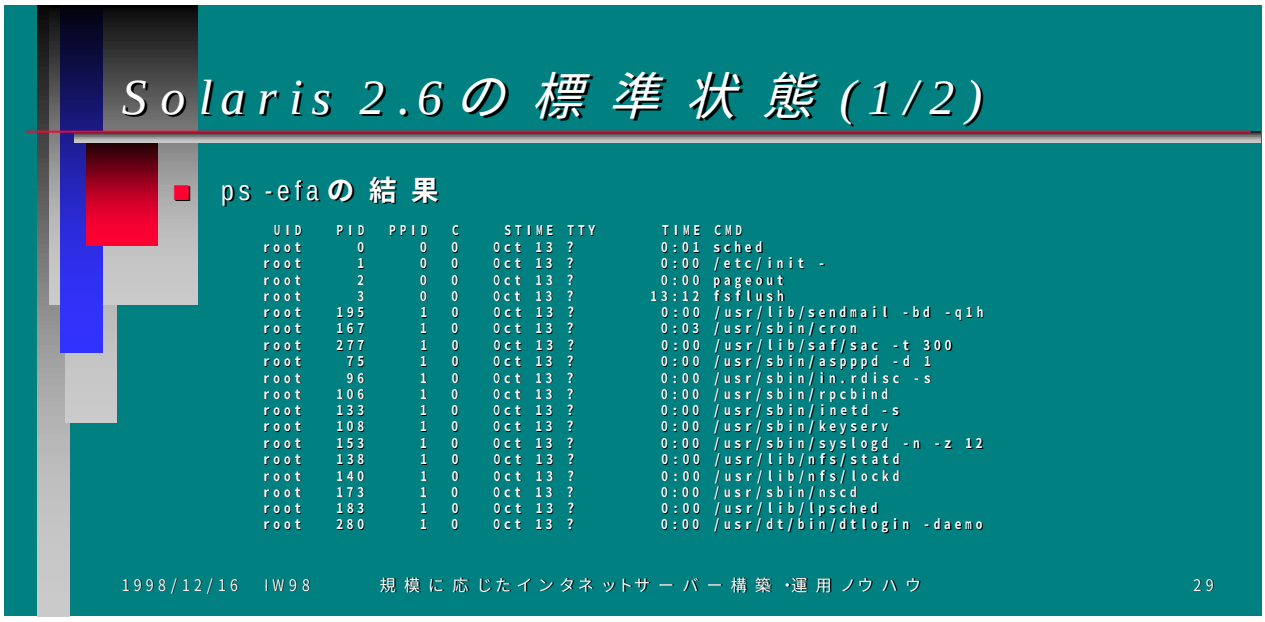
3．Operating System

OSのチューニング

OSのチューニングについて説明します。Sunをインストールすると様々なdaemonが動きます。実際に使わないものまで実にいろいろ動いています。「与えられたまま使っていては負けである」と私は言っていますが、使わないものは動かさないというのが基本です。SolarisでpsをとるとⅢほどプロセスが動いていて驚きます。 これに対してFreeBSDはそれほどdaemonは動かないので、あまりチューニングの余地はありません。ほとんど動かないプロセスでも、そのプロセスを止めることによってメモリの節約になったり、プロセステーブルの節約になってトータルスループットが少し向上しますので、検討する必要があります。「動いていないからほっておいてもいいや」というのではチューニングにならないのです。

Ｓｏｌａｉｓ ２．６の標準状態

下図はSolaris2.6、Sun OS 5.6の標準状態でのps結果です。 ６０数個のプロセスが動いています。 OpenWindowsのdaemonもありますし、システムに必要なdaemonもあります。私は、こういう場合、要らないものは止めるという作業をします。



停止したサービス

インターネットサーバーの場合は、デスクトップは使わないのでOpenWindows関係は必要ありません。ですから、セッションマネージャ（dtlogon）は止めます。仮名漢字変換も必要はないので、仮名漢字変換のサーバー類（WunとATOKと２種類）も勿論止めます。それ以外にもkkcvも止めます。インターネットサーバーの場合は、ごく一部でNFSも使われますが、スタンドアローンで動かす場合が多く、NFSは不要なのでNFS関係も止めます。NISも使いませんので、やはり止めます。RPCですが、Sun OS 4.xや最近のFreeBSDではportmapというプログラムが動いています。（ Solarisの場合はrpcbindというプロセスです。） これを止めたいのですが、実は簡単には止まりません。inetd.confにrpc....というプログラムが沢山動いていますが、これらを全部コメントアウトしますと、rpcbindを止めても問題が発生しなくなります。

vold（volume management daemon）はCD-ROMを挿入したときに、自動的にマウントしてくれるdaemonですが、インターネットサーバーとして使っている場合は不要なので、止めてしまいます。snmpは使う人もいると思いますが、私の場合はあまり使わないので止めてしまいます。プリントアウトも要りません。

サーバーによってはsendmailを止める場合もあります。といいますのは、最近いろなところで目につくSPAM（不正メール）を防ぐにはsendmailを動かさないのが一番簡単な方法なので、要らなければ止めてしまいます。ただ、システムのクローンのメールは注意が必要で、メールを出した瞬間に相手のところに届けば良いのですが、そうでなかった場合に急に落ちてしまうこともありますので、ベストなのはsendmail -q30mにして、-bdというオプションを抜いておく事だと思います。 そうすると、sendmail自身はメールを受けることをやりません。 つまり、SMTPの受け付けを一切行わずに、qに落ちているメールの再送のみをやってくれますので、-q30mとやっておくのが良いと思います。実際、私の持っているサーバーの何台かはこの形にしています。-bdだけは抜いて、そのかわりsendmail cfはそのままにしておきます。

Ｓｏｌａｒｉｓ２．６のチューニング後

様々なプロセスを止めた結果のpsが上図です。先程説明したbdが残っている図と比較すると違いが良く判りますが、これ位までは止める事ができます。昔、24Mのメモリしか持っていない（今から見るとものすごく非力な）ＳＰＡＲＣ IPCワークステーションをこのような状態にして様々なソフトウエアテストをやっていたのですが、実際に使えていました。 ここまで止めるとやはりSolarisというのは思ったほど重くないなというのが私の実感です。

サービスの止めすぎに注意

止める場合に一つ注意しなければならないのは、止め過ぎです。使っていないと思って止めてみたはいいが、/usr/bin/OpenWindowsとキーボードから打込んだ瞬間、起動するまでに異様な時間（数分）がかかるという状態が起きます。これは主に日本語関係、あるいはメール関係の余分なものまで止めてしまったことが原因です。この間もNetscape Enterprise Serverが起動しなくなりました。動かなかった原因は、nscd（ Name Service Cache Daemon）というプロセスだと思われます。プロセスID136番にあるnscd（ホストネームやユーザー名のキャッシュをやってくれるらしい）を止めたら、Enterprise Serverが動かなかったのです。止め過ぎには注意しましょう。 止め過ぎの原因が判らないときは、結局１個ずつ動かしてみては、これじゃない、これじゃないと言って探す事になります。サービスの中には止めるとパフォーマンスを落とすものもあるようです。これは推測ですが、nscdを止めるとEnterprise Serverのパフォーマンスが落ちるのだと思います。

ＦｒｅｅＢＳＤのGENERICカーネル

FreeBSDをインストールした直後のカーネルはGENERICなカーネル、要するにインストールがうまくいくような最大公約数的なカーネルになっています。 最大公約数的ですから、SCSIコントローラのドライバが10種類以上もあつたり、いろいろな種類のネットワークインターフェース用ドライバも入っており、CD-ROMドライバ関連も最近使われ

ていないミツミやソニーや松下等の専用ドライバ、等も入っています。ですから、これを削ることによってカーネルはシェープアップする事ができます。カーネルの標準に入っている内部のテーブル類は非常に小さいです。多くの場合はそれでも困らないのですが、激しいサーバー、ヘビーなサーバーで使う時には内部テーブル類もきちんと変更しなければなりません。

リコンフィグ

必要なものだけで、カーネルのリンコンフィグ作業を行います。 私は、NFSは抜いてしまいますし、サーバーとして使う場合はフロッピーやDOS用のCD-ROMも使うこともないですから、ドライバを抜いてしまいます。あとはネットワークインターフェースも実際にハードウエアに実装されているものだけを残して、それ以外は削ってしまいます。あともう一つは、テーブルを必要な大きさに増やすいう作業です。私の場合、maxusersは迷わずに256というように大きくします。

デフォルトのmaxusersは10で、usersを10人ぐらい想定したときのいろいろなテーブルのパラメータがを決める為に使われます。 これを増やすとカーネル内の様々なテーブルのサイズが大きくなりますが必要なものは増やします。ネットワーク関係ですとNMBCLUSTERSというカーネル内のMBufの数を決める項目があります。 FreeBSDの場合、maxusersを増やすとNMBCLUSTERSも増えます。多くの場合はデフォルトで困らないと思うのですが、必要ならば増やします。

次は、FD_SETSIZE(標準は256)ですが、これをきちんと大きくしないと、一つのプロセスで大量のネットワークコネクションを張るプログラムの場合にはうまくいきません。

FreeBSDの場合ですと、LINTというファイルが/sys/i386/confの下にあります。隣にGENERICというファイルがありますから、LINTのファイルを見ながらGENERICを修正すると便利です。

FreeBSDのカーネル

FreeBSD 2.2.7-RELEASE **で** GENERIC **カーネルとリコンフィグ後のカーネル**

```
$ ls -l kernel.GENERIC kernel
-rwxr-xr-x  1 root  wheel  1564800 Aug  5 03:57 kernel.GENERIC
-r-xr-xr-x  1 root  wheel   764284 Sep 29 08:45 kernel

$ size kernel.GENERIC kernel
text     data     bss      dec      hex
1294336 81920    91112    1467368 1663e8  kernel.GENERIC
589824  53248    51600    694672  a9990   kernel
```

上図には実際のリコンフィグ後のサイズが書いてあります。もともとのカーネルというのはkernel.GENERICという名前で、1.5Mぐらいあります。これをリコンフィグすると、764Kぐらいまで減ります。sizeコマンドで見ますと、テキスト領域もデータも減っています。

Solarisの場合

Solarisはカーネルのソースが公開されていませんので、FreeBSDのように簡単にリコンフィグする事はできません。 デバイス類はboot -rオプションを指定すると必要なデバイスのみロードされるので、デバイスドライバが多過ぎるような状態にはなりません。必要ならば、FD(ファイルディスクリプタ)を増やしたり、ptyを増やしたり、共有メモリのサイズを変更します。 ptyはSolarisではデフォルトで48しかないので、ウィンドウをたくさん開いてしまうとptyが足りなくなります。

このような情報は、Sunの配布ドキュメントの中に少ないのですが、AnswerBookの中に一部あるのと、Solaris-ⅡFAQを参考にしています。

/etc/systemで、ファイルディスクリプタの増加、ptyの増加、共有メモリの変更、100Mのネットワークカード(Etherインターフェース)をfull-duplexにするのか、half-duplexにするのかという選択ができます。Solarisのチューニングで、もしネットワークパラメータをチューニングしなければならない状態に陥った場合は、/dev/tcpというデバイスがありますので、nddコマンドを使ってパラメータを変えることができます。変数の種類は、nddコマンドで/dev/tcp ?とやりますと出てきます。

dev/tcpなのでipやdev/udpもあるのですが、一般にはほとんど変更しません。

/etc/rc2.2/s69inetを見ますと、/dev/ipもありますが、これもほとんど変更しません。変更が必要になるのは特別な場合だけです。

４．Server Software

WWWサーバー

現在ではインターネットの代名詞ともいわれるWWWです。WWWサーバーはご存じのとおりHTTPのリクエストを処理するサーバーです。インターネットの中で最も稼働台数の多いサーバーだと思います。 といいますのは、１つのサイトで、ネームサーバーですとせいぜい２つ位で良いのですが、WWWサーバーですと10台位の数になります。マイクロソフトのWWWサーバーに至っては何十台という数になります。サーバーのチューニングを行う場合に、WWWサーバーがどういう動作をしているかをよく観察する必要があります。 HTTPの処理を簡単に説明しまと、TCPが接続されてから、クライアントのリクエストが送られてくると、それに対してサーバーがコンテンツデータを返し、最後にTCPが切断されるという手順になります。 TCPの接続切断が大量に起きるのです。

WWWサーバーレスポンスまでの処理

レスポンスまでの処理は、まずリクエストを解析して、ページの読み出しであるのか、CGIの実行なのか、あるいはページがURLでmappingされているのか、を判断し、ページデータの読み出しやCGIを実行します。ログの生成もWWWサーバーの重要な仕事です。ログを生成する場合に、必要ならIPアドレスからFQDN（Full Qualify Domain Name）を知るために、ネームサーバーへの問い合わせを行うこともあります。必要ならと書きましたが、DNSの問い合わせは時間（長い時だと１秒以上）がかかりますので、WWWサーバーでよく言われるボトルネックの一つになっているからです。 最近のApacheでは、デフォルトがこの問い合わせはしない設定になっています。Netscape Enterprise Serverもこれがオン／オフできるようになっています。 （実際のログの生成は、レスポンスを返した後に行われているようです。） そして、クライアントへのレスポンスを返すのです。

WWWサーバーボトルネックの要因

ボトルネックの要因を挙げてみました。まずコンテンツデータの読み出しですが、Diskの

I/Oが入りますのでボトルネックの要因になります。読むためには、そのプロセスはどうしても待たないといけないので、１つのリクエストについてディスクが応答するまでの時間がかかるわけです。あとはCGIプログラムの実行です。これが実はくせ者でして、CGIは多くの場合、外のプログラムを呼び出します。ですから、execの時間がかかったり、CGIプログラムが大きかったりするとメモリが不足してスワップが起きることもあります。
それと、よく言われているCとPerlの問題です。 CGIが、Ｃで書かれているか、Perlで書かれているかが影響を与えるのです。Perlは非常に便利な言語であって、文字列の処理も速いのですが、起動はそんなに速くありません。ですから、Perlで書くとトータルのスループットは落ちることがあります。
もう一つ、本当にヘビーなサーバーになってくると実際に起きるのが、ログ書き出しの負荷です。 例えば１日に100万ヒットもするようなサーバーですと、10行のログを１秒間に書かなければならないのですから、ログファイルも無視できない。先ほど述べたDNSを調べてFQDNで記録する場合も、ログの記録が遅くなる原因になります。 ネットワークロケーションが問題になる場合もあります。

WWWサーバー

1 requestの負荷

WWWサーバーの１リクエスト当りの負荷は、「リクエスト数×処理時間」で計算できます。
例えば１リクエストに１秒かかって、メモリを１Mを使用し、平均で１秒間に10リクエストが来る場合、この処理にはメモリが10M必要になります。これが毎秒100リクエストあるのだとすると、メモリは100M必要になってしまうのです。もちろんこれは単純な計算なので、実際はもっと悪い状態になるかもしれません。リクエスト数つまりプロセスが増えることにより処理の手間が増加しますから、実際の負荷はもう少し増えてしまいます。
ところが、CPUが10倍速かった場合には、処理時間0.1秒になります。処理時間が10分の１になりますから、100リクエストあったとしてもメモリは10Mで済むようになります。
もう一つ忘れてならないのは、ネットワークの転送スピードです。CPUを10倍に速くしてもネットワークは10倍速くならないのです。WWWサーバーの場合必ずクライアント側のスピードも関係します。転送速度が遅いとサーバーでのプロセス存在時間も長くなるからです。いくらサーバー側のCPUを速くして、ネットワークを速くしても、相手が常にダイヤルアップの場合だと１プロセス当たりの処理時間は変わりません。こういう点も考慮しなければなりません。

WWW Server Software

WWWサーバーでメジャーなソフトウエアと特徴を次に示します。

・CERN	リクエスト毎にfork
・NCSA httpd	高機能でCERNより高速
・Apache	fork済みのプロセスにfd passing　NCSA httpdも採用
・phttpd	ｆｏｒｋの代わりにthreadを採用
・thttpd	selectを使ったループで処理

CERN Server

最近はもうほとんど使われていないのですが、CERNというサーバーが昔ありました。WWWサーバーのサンプルインプリメントでして、プロキシサーバーとWWWサーバーが１個で兼用できたのです。その点が便利で一時期広く使われていました。ただ、これの欠点はリクエストを受け付けてからfork処理を行ってレスポンスを返すという構造ですので、

forkの処理が重くパフォーマンスがあまり上がりませんでした。今は他に良いソフトウエアがあるのでほとんど使われなくなりました。

NCSA httpd
NCSA httpdは、CERNよりも機能が高かったために一時期広く使われていました。URLのリダイレクト、アクセス制限の設定が柔軟だったところがよかったのです。実際に計測しましたが、CERNのサーバーよりは高速でした。

Apache WEBサーバー
現在世界で一番可動台数の多いWEBサーバーが、ご存じのApacheです。Apach開発者の話によりますと、正確な動作を重視したインプリメントを行ったということで、スピードよりも正確に動作することを優先しています。といっても、十分に高速です。Apacheもforkを行いますが、リクエストが来てからforkしているのでは遅いので、あらかじめforkしたプロセスを複数個用意しておいて、それぞれのリクエストに応じて子プロセスを順に起動するような形にしています。ですから、リクエストごとにforkをしない分だけ少し速いわけです。また、バーチャルホスト対応でもあります。

Phttpd & thttpd
あまり知られていませんが、特徴を持っているのが、**phttpd**と**thttpd**の２つです。
phttpdはスレッドを利用してリクエストを処理します。スレッドはforkに比べるとはるかに軽いために、非常に高速な処理が可能になります。実際にNetscape Enterprise ServerのSolarisバージョンは内部でスレッド処理をしています。ただ、スレッドがきちんと動くOSが少ないので、移植性に多少難があり、現状はSolaris以外で動いている例はありません。
もう一つ面白いのがthttpdで、私はこれが好きでよく使っています。複数コネクションをselectでポーリングをかけながら同時処理しますので、一切forkは起きません。その関係もあって非常に高速です。しかしながら、httpd1.1プロトコルはちゃんと満足しCDIはできるのですが、機能的には非常にプリミティブでURLのリダイレクトをしたいと思ってもできません。

NEWSサーバー
NEWSサーバーは私の知る限り最も負荷のかかるサーバーです。特にISPのバックボーンで使いfull feedすると、１日に90万通位の記事がやってきて、20Gbyte位のトラフィックを生成します。NEWSサーバーは他サーバーとの記事交換と、エンドユーザー向けの購読、投稿サービスを行つております。

NEWSサーバー
記事の受信
NEWSサーバーは記事を受信する場合、自分が持っていない記事を選択して受信するという構造をしています。まず他のサーバーから記事を受信すると、記事の重複の検索を行います。これはヒルトリールックアップと呼ばれるもので、ヒストリーというメッセージIDベースのデータベースを持っていて、受信済み記事か否かの判断ができるようになっています。その後、スプールへのデータの書き込みを行い、別サーバーへ転送が必要な場合には、送信ファイルの生成をおこないます。ヒストリールックアップはデータベースを検索するので、readのDisk I/Oを頻発します。最近のトラフィックは、T1でも不足気味でfull feedを送る事ができない位のボリュームがあります。

NEWSサーバー
記事の送信
記事の送信は、記事を送信ファイルから読み込んで送信するという処理を行います。これも配送先が１個なら軽いのですが、20や30と多くなると、スプール読み込みの量が増えて、やはりボトルネックになります。この実例はあとで紹介します。

NEWSサーバー
ユーザの購読用

ユーザー購読用サーバーは、管理する情報が増えます。 ニュースグループと記事番号の管理、overview情報、history情報からスプールの記事への対応、cancel処理の問題もあります。

従来のＩＮＮ（ｂｅｆｏｒｅ１．ｘ）

NEWSサーバーソフトウエアとして有名なものに**INN**(私はアイ・エヌ・エヌと呼んでいますが、インと呼んでいる人もいます)があります。 従来のINNはニューススプール形式にufs（UNIX File System）をそのまま使っていました。これは記事番号のファイル名で送るという形で、ディレクトリはニュースグループのディレクトリです。news.software.nntpの350番目の記事ですと、/var/spool/news/news/software/nntp/350というファイル名で、パス名のところにファイルの実体があります。この形式では、ファイル数が増えた場合、nntpのディレクトリに置かれるファイル数が一杯になります。コントロール記事の処理が/var/spool/news/controlのディレクトリに置かれている関係で、cancel（記事の取り消し）処理が来ると /var/spool/news/controlまで行ってファイル削除を行うので、コントロールの記述が膨大な最近のニュース量では、ufsがボトルネックになります。

Ufsの問題

ファイルをつくる場合を考えますと、ディレクトリ内に同じファイル名が存在しているかどうかをリニアサーチしてみなければなりせん。ファイルが１万位になりますと、サーチするのにかなりの時間がかかって、無視できなくなります。ファイルを消す場合は少し楽ですが、それでも目的のファイルが見つかるまで順に探さなければなりません。先ほど説明したコントロールメッセージは今１万ぐらい同じディレクトリ内にありますので、その特定のファイル、コントロール記事の書き込みにボトルネックが発生します。

Diablo

最近あらわれたNEWSサーバーのソフトウエアに、Diabloがあります。 もともと配送専用のサーバーで、ISPのバックボーンとして、隣から受け取った記事を更に隣へ送るという機能を高速に行うよう作成されています。高速の理由は、キャンセル処理をしないので、余計なDisk I/Oが発生しませんし、activeやnewsgroups等のファイルが（必需品ではあるのですが）が無いからです。しかし最近、Diabloにも購読用サポートが入りましたので、現在はactiveがあります。当時の話ですが、INN 1.xの５～10倍のパフォーマンスがありました。 これは、ニューススプールを独自の形式にしたからです。INNのバージョン１の場合、１つの記事が１個のファイルになっていてディレクトリに大量の記事が置かれるのですが、Diabloの今のバージョンでは、時間と関係したディレクトリの中にニュースの記事の巨大なファイル（100個から1,000個）をつくるので、非常にパフォーマンスが上がっています。

現在のＩＮＮ（version ２．ｘ）

INNもバージョン2.0からCNFS(Cyclic News File System)を導入しました。あらかじめ巨大なファイルをつくっておき、頭から順に使っていき、最後まで行ったらまた前から使うという使い方をしています。cancel処理が劇的に改善されていますので、Diabloと変わらない位のパフォーマンスになっていると思います。

Mailサーバー

メールサーバーには２種類のMailサーバーがあります。一つは**メーリングリスト用のメー**

ルサーバーで、速く大量に、より多くのユーザーにメールを配るというのが目的です。
もう一つは、 **POP3、SMTPサーバー**と呼ばれるPCユーザーのダイヤルアップ用のサーバーで、１台のマシンでより多くのユーザーをサポートできる方が望ましいわけです。

メーリングリストサーバー
メーリングリストサーバーの**sendmail**は配送が非常に遅かったので、ある記事の配送が終わる前に別の人が返事を書く可能性がありました。sendmailでの従来の問題点です。最近それを改善するために**qmail**や、**sendmail＋WIDE patch＋smtpfeed**が比較的多く使われています。

Sendmailの特徴

従来メールサーバーというとsendmailしかありませんでした。 sendmailは基本的に逐次配送しか行いません。sendmail＋WIDE patchというのが昔のバージョンです。 WIDE patchを当てると少し速くなるのですが、それでも配送部分については順にしか配りません。

逐次配送ですと、メールが送り先に届かないケースが発生した場合、それを判断するまでに少し時間がかかります。 また、ネットワークの状況が悪くてパケットロスが起きる場合にはTCPで再送が行われて、時間がかかります。1,000人位の配送先があった場合、途中に少し遅メールが数個入っているだけで、最後の人のメールに対してはものすごく遅くなります。最初のメールが投げられてから最後のものが送り終わるまでに、３時間もかかかったりします。sendmailの良い点は、同一MX先の相乗りができる事です。

user1@foo.co.jp,user2@sh.foo.co.jpのように、foo.co.jpとsh.foo.co.jpが同じMX先のホストであった場合は、１通の配送で終わります。もし100アドレスあったとしても、MXの先が１つだとすると、投げるのは１回で済むので速いのです。

qmailの特徴

最近話題になっていますqmailです。これはもともと高速配送という目的ではなくて、sendmailのsecurity holeがあまりにひどくて、インプリメントもこれではいけないと思った人がつくったプログラムで、複数の小さなプログラムを組み合わせて用途別に処理するのです。

qmail-smtpd、qmail-inject、qmail-remote、等の細かいプログラムが一杯あります。昔sendmailはsecurity holeが数多くあり、しょっちゅうバージョンアップしなければならなかったものの一つだったのですが、最近はあまりsecurity holeは出ていません。

qmailの面白いのはsecurity holeはないぞと威張っていまして、security holeの発見に1,000ドルの賞金がかかっています。処理的にはsendmailよりはるかに軽いし高速です。ただ、同一ドメインであっても１通ずつ配送するという問題があり、先ほどの例ですと、usr@foo.co.jp、usr1@foo.co.jp、usr2@foo.co.jp……と100種類のアドレスがあった場合には、100通のメールを投げてしまいます。 制限はできるのですが、それぞれのあて先に対して１個のプロセスが起動しますので、同時に100個のプロセスが動くという事態が発生します。DiskI/Oが絡んできますので、ハードディスクの構成をよく考えないと、ディスクの読み出しでボトルネックになることがあります。

Sendmail + WIDE Patch + smtpfeed

sendmail（例えば8.9）＋WIDE patch＋smtpfeedという組み合わせが最近の高速なメール配送ではよく使われている手段です。smtpfeedは、sendmailが１通ずつ逐次配送するのを改善するために、外部mailerとしてsendmailから呼び出され、実際の配送を分担します。ですから、アドレス等の解析は行わずに、sendmailからもらったアドレスに対して投げるだけで、sendmailとLMTP（Local Mail Transfer Protocol）というプロトコルで通信します。

この組み合わせの良いところはMXの相乗りを行います。つまり、selectを使って複数に同時に配送を行うのです。Aのアドレスにメールを投げている間に、Bのアドレスに対してもメールを配送します。おかげで、極めて高速な動作が可能です。実際に、DX4でやってみたのですが、1,600アドレスを３分以内に90%のメールを配送しました。非常に高速です。もちろんこの数字を出すためには、ネットワークのロケーションもそれなりによくする必要があります。従って、これは１個のプロセスで全部処理します。さき程のqmailの場合は、アドレス数だけのプロセスが上がりますので気をつけないと、プロセス制御やメモリの問題が出てきます。

smtpfeedはすべてをメモリ中で処理しますので、宛先が増えるとメモリの使用量も増えます。

POP3、ＳＭＴＰサーバー

POP3とＳＭＴＰサーバーはエンドユーザー用のサーバーです。SMTPは普通のメールサーバーと変わりません。
POPサーバーには、一般に**qpopper**というプログラムがよく使われています。
/spool/mailにユーザ数分のファイルが置かれ、アカウント数が多いと同一ディレクトリに大量のファイルが作られるため、ファイルを探す際に、先ほど説明したufsでリニアサーチする事なります。

qpopperの注意点

更に悪いことに、qpopperはデフォルトのままですと、リクエストがあるとメールボックスを１行ずつ同じディレクトリのテンポラリファイルにコピーし作成します。１万ユーザーあると、そのテンポラリファイルの分も含めて２万個分のディレクトリエントリを消費するので大変です。さらに、接続終了時に元のファイルに書き戻すので、ufsによるディレクトリのリニアサーチの問題も効いてきます。ただしqpopperはソースが公開されていますので、テンポラリファイルを別のところに置く等の改造をするとパフォーマンスを上げる事ができます。

ＯＳのインストール

先ほどのチューニングではなくて、これはOSインストールの基本部分についてです。必要なものだけをインストールするという考えを基本にしたほうが良いと思います。余計なものをインストールしますと、ディスクを消費します。 Solaris(2.6)の場合、CDEまでインストールされてしまいます。昔よく言われていた事ですが、「swapサイズはメモリの倍とる」というのがありました。 けれども、これは今となっては過去の話です。
例えば、メモリを512Mでメモリの倍の１Gのswapを切ったとしましょう。これを動作させた時に１Gの中でswapを書き込むことを考えるとんなに速いディスクを持ってきても１分ぐらいはかかってしまうと思います。swapが起きると著しいパフォーマンスの低下になってしまうので、こんなに多くのswapを切る必要はないのです。私は、十分なメモリを用意したらswap用のパーティションは切らないようにしています。Solarisの場合はswap0にするのはインストーラで簡単にできます。FreeBSDの場合はswap0とするとインストールできないので、少し工夫が必要です。この間、swap領域を１Mと書いてみたら、ファイルシステムとして認識してくれないようでうまくいきませんでした。FSタブからswapの部分をコメントアウトしないといけません。
従って、メモリを用意したらswapは0でいいのです。これがOSのインストールの注意点です。

運用開始後の監視
次は運用開始後の監視です。インターネットサーバーの場合は、必要とするレスポンスが十分に出ていれば、ぎりぎりでも良いというのが基本にあります。さまざまなCPUの状態を監視するプログラムがありますので、それらを利用して定期的にチェックすることが重要です。

５．Real Example

実際のサーバーの例です。

news.nspixp.wide.ad.jp[1996/10]
先ほど説明しましたNEWSサーバーです。Nspixp（WIDE Projectの商用IX、Internet eXchante）のNEWSサーバーがあります。（実際にGNSを引いていただくとすぐわかります）
ニュースの交換というのは基本的には１対１で交換しなければいけない（ 例えば３つのプロバイダー、A、B、Cを考えた場合に、それぞれニュース交換しようとすると、AとB、BとC、CとAがそれぞれnmtpの設定をしないといけない）のですが、IXのような場所ですと、数が増えるとメッシュ状に増えてしまって大変なので、間にマシンを１台置いて、そこと交換すれば良いのではないかと考えて始めてみました。
1996年の秋に作成したサーバーで、構成は次の通りです。
・Pentium5 133MHz。メモリ 128MB
・HD ２Gを２つと４Gを１台

２Gは１台をシステム、１台はニュースのhistory、４Gのディスクをスプールに置きました。SCSIは、先ほど説明しましたように同時に書き込めるというパフォーマンスを期待して、２G２台のほうに１チャンネル、スプール専用に１個を使ってみました。最初はさすがに良いパフォーマンスで動いていたのですが、時間が経つに連れていろいろと問題が起きました。ソフトウエアはDiabloを使いました。しばらく運用していると、ニュースが送れなくなったのです。何が悪いのだろうと、メモリ、ディスクを調べていたのですが、最終的にCPUがネックではないかと考え、vmstatで調べたら50%余っている。 でもどうも変だと思い、CPUをPentium-Pro200MHzにかえたところ、速くなりました。
その他、netstatの結果で、incomingが多いとoutgoingが減っているし、outgoingが多い時はincomingが減っている。 合計するとEtherの帯域一杯ではないかという事になり、調べたところ、100M対応のインターフェースを使っていたのですが、Ether Switchの設定が間違っていまして、
１００Mのつもりが実際には10Mで通信していたのです。設定を変更したところ、一回りパフォーマンスが上がりました。つまりEthernetがボトルネックになっていたわけです。

freeBSDでのチューニングオプションとして重要ですが、mount optionにnoatimeがあります。
UNIXはファイルをopenしてcloseするだけでも、アイノート上でアクセスタイムの更新を行います。
この数が１個や２個なら良いのですが、NEWSサーバーのようにスプールを大量にアクセスすると、アイノートの更新時間が無視できないので、noatimeを追加してみました。そうすると、観測した限りは少しだけパフォーマンスが上がりました。この時はほんの少しです。もっと強烈なオプションにasyncがあり、できるだけディスクに書き込みに行かないというオプションです。

asyncの状態でシステムがクラッシュするとファイルシステムに膨大なダメージが起きると思いますので、普通にはお勧めできないのですが、ニュースプールだったら大丈夫と考えてやってみました。 これは全然効果がありませんでした。 まだメモリが128Mの時です。 asyncで書き込みを遅らせたところで、結局バッファリングに使えるメモリがあまり余っていなかったので効果がなかったわけです。
この後、Diabloで**shm**（シェアードメモリ）を使うように修正してみました。Shmはニュースの**SPAM(**あちこちの記事に大量にポストする) のチェックに使っていたと思います。configのミスでshmがうまく使われていなかったのを使えるようにしたのですが、かなり効きました。DiabloはSPAMの情報をハードディスクに書くので、shmを使うと大きな差が出ます。incomingは12G位だと１日当たり六十何Gぐらいのoutgoingを出すだけの能力はあったのですが、incomingが18.57GB/日になってしまうと、outgoingが減ってしまうのです。シークが非常に増えてスプールのボトルネックが発生するのです。 ２時間に１回expireをかけていた記事の古いものを捨てるのですが、それでもものすごい量で、こちらへ来た記事をよそへ配る前にexpireによってなくなってしまうという事が発生し、outgoingが４３．１７GBに減ってしまったのです。その後、メモリを増やしてみたら、expireで当初30分位かかっていたのが、バッファリングが効くようになり半分に短縮できました。ディスクバッファを大量に必要とするようなプログラムですと、メモリをたくさん積むのが有効です。

news.nspixp2.wide.ad.jp[1998/1]
先ほどのマシンがあまりにスプールが小さくて、もっとスプールのI/Oを稼がなければならないというので、作ったのがこのマシンです。98年１月に作成しましたので、ちょうど１年ぐらい前です。
・Pentium-Ⅱ 333MHｚ
・メモリ 384MB
・ HD ９Gを２台

CPUは、発熱が少ないのと入手し易さを考えて選択しました。発熱というのは結局マシンの寿命を縮めるもとですので、発熱の少ないものにこしたことはありません。このCPUは、当時はまだ出たばかりだったのですが、今後を考えると入手しやすくなるだろうと判断しました。
というのは、CPUが壊れたときは交換しなければいけないので、運用を考えると入手しにくいCPUを使うというのはよくないのです。メモリは、先ほどのNEWSサーバーが256MBまで載せたのですが、どれより多くのメモリが必要だということで、この時のマザーボードの限界、128MBのDIMMを３枚、まで載せて384MBにしました。スプールのI/Oを稼ぐために当時の９Gのディスクを、ccdを使ってストライプにしました。接続先ポートの関係でFDDIインターフェースを使っています。

news.nspixp2.wide.ad.jpの実力
31のISPと接続され、１日当たりのトラフィックは、ｉｎｃｏｍｉｎｇの記事数が55～80万通、18～23GBというとてつもなく多い量です。この数字が出てくるためには、２Mbpsぐらいないとだめです。
つまり、T1では送り切れないと言われている量です。１日に330GBのoutgoingの能力を持っています。12月６日に最大トラフィックが発生しましたが、１日にincomingが25.7GB、outgoingが361GBになりました。普段は、FDDIの40%ぐらいの帯域をコンスタントに使っていますが、ピーク時はFDDIの帯域を使い切っています。
必要であればこういう大規模なマシンをつくれば良いのですが、このマシンは最近バテぎ

みで、先ほどお見せした破綻を時々起こしています。

sh.janog.gr.jp[1998/8]
JANOGのメールサーバーとして最近（98年８月）つくりました。sh.janog.gr.jpです。実稼働を始めて3カ月ぐらいですが、JANOGのメールサーバーとWWWサーバーです。
・CPU　DX40DP100
・メモリ　　１６MB
・HD　SCSI 2G　１台
・MB　ISA/VL
・OS　FreeBSD2.2.7 RELEASE
・www　thttpd 2.04
・mail　　sendmail8.9.1a + WIDE patch3.1W + smtpfeed0.90

このシステムはJANOG内で相談して、NTTPCのハウジングサービスにマシンを置くことになり、お金をかけずにつくりましょうという少し貧乏くさい設計目標のもとに作りました。
ですから、ごみになる直前のDEC PCを拾ってきました。CPUはDX66だったので、少しでも早くなればと思いDX4ODPをどこかから拾ってきました。メモリは最初に拾ってきた状態で16Mだったのですが、OSが１Mぐらい、動かすプログラムを考えてもりぎり足りるだろうということで、16Mのままです。ハードディスクはSCSI１台で２Gを入れてあります。SCSIのコントローラはおなじみのAdaptecの1542CFというものを使っています。当然こんな時代ですから、マザーボードはISA/VLです。

JANOGのメールを受けていらっしゃる方、最初の人がポストしてからお手元に届くまでが結構早いと思うのですが、現状５分ぐらいでほとんどのメールを配り終わっています。メモリ16Mで現状十分なのは、smtpfeedはピーク時には２Mぐらいメモリを使うのですが、ふだんはsendmailしか動いていなくて、sendmailが待っているだけだからです。thttpdも１Mしか使いませんし、コネクションが増えても使用メモリがそれ程増えるわけではありません。また、forkも行わないのですから、ほとんどメモリを圧迫しません。
何が一番圧迫するかと言いますと、チェックに入ってログインする際にウィンドウをいくつか開くと、バッシュやTCシェルの方がメモリを圧迫します。ふだんはサーバーですからログインする必要もないので、その分のメモリは空いています。現状は16Mで十分です。それでも、現在1,660人ぐらいのメールアドレスに配っています。まだWWWサーバーとしてはあまり活躍していませんが、いいパフォーマンスを示しています。

COTURA AERO 4/33C [1998/11]

私が自宅で個人的に使っているサーバーです。余り物を使うのは好きでして、COMPAQのCOTURA AEROです。これはつくったのはつい最近で、98年11月となっています。私の場合、自宅では単なるダイアルアップユーザーなので、ごみになったサブノートパソコンを使っています。

・CPU　SL486SX/33MHｚ
・メモリ　　12M
・HD　IDE840M(もともとは250M)
・OS　FreeBSD2.2.7 withPAO

自宅でのWWWサーバーとproxy用です。家の中ではローカルIPアドレスを使っていますから、このマシンを経由してインターネットへアクセスするので、proxyサーバー等のリレーするツールを運用する必要があり、WWWとproxyサーバーでいくつか実験してみました。

squidは一般にすごく有名で、高速で、確かに非常に速いのですが欠点もあります。 psでずっと観測していたのですが、squidを動かすと５M以上メモリを食うのでほかのプロセスの挙動に影響しますし、何も使っていないときでもコンスタントにセレクトのチェック等を行うのでCPUを消費します。最近のマシンですと、squidは定常状態のCPU負荷位は無視できるのですが、こういう遅いマシンでは無視できないという例です。これがあまりに気になったので、apacheにmod_proxyというモジュールを加えてみました。apacheはアクセスされなければ余計なCPUを食いませんので、メモリの使用量はsquidに比べて少なくなりました。 しかし、apacheを動かしてしまうと、結局apache自身がWWWサーバーにもなるので、thttpdはやめたのですが、それなりのスピードは出ていました。もちろんproxyサーバーとしてはsquidのほうが明らかに高速です。

Solaris 2.xでの例

今度は先日作成したサーバーの例です。 もともとUltra1（メモリ512M）のマシンがありました。ここで１Gメモリを欲しいという要求があったのですが、1Gのメモリは高価です。何でそんなにメモリが要るかと言いますと、このサーバーはメモリ512で、60Mや70Mの数個のプロセスがぎりぎり動いているのですが、もう少し何とかしたいのでメモリを増やしたいというのです。

CPUがUltra1（UltraSPARCの167MHｚ）ですが、このCPUにメモリを１Gも積んでも大丈夫かと考えたのです。大丈夫かというのは動く動かないという問題ではなくて、投資に見合うかというところを検討しないといけないわけです。

CPUの能力に対してあまりに巨大なメモリを載せても、バランスが崩れていると感じるわけです。考えなければいけないのはこの事です。

Solaris2.xでの例　SPARCengine UltraAXiの導入

これではよくないと思い、いろいろ調べた結果、SunがOEM用に出しているマザーボードを見つけました。これはSunのOEM用プロダクトでして、OEMですから紙に印刷されたマニュアルはなく、PDFのファイルがあるだけです。そのマザーボードは、ATXのボードサイズに300MHｚのUltraSPARC-ⅡIで、現在のCPUに比べて一回り高速になります。これ位のCPUであればメモリを１Gぐらい積んでもバランスがとれると私は思いましてた。ボードだけなので、自分でケースも買わなければいけないし、もちろんメモリもきちんと選ばなければなりません。

オンボードにネットワークインターフェースとSCSIインターフェースが付いています。（今売っているUltra5にはSCSIインターフェースはありません）ですから、Ultra10より

も良いのではないかと思います。このシステムはサーバーですから画面はどうでも良いので、秋葉原で売っている7,000円ぐらいのPCIのビデオカードで使えます。モニタも買う必要はなかったし、CD-ROMドライブはインストールのときしか使わないので買わない。ラックマウントのケースにしておくと後々便利だと思ってラックマウントのケースを買い、大体100万円位で作成できました。もちろん自分で組み立てなければいけないので、PCを組み立てられない人やSunに詳しくない人にはお勧めできませんが、Sunを長く使っている人だったらこういう方法もあると思います。こういう導入の仕方というのも考えられるのです。もちろん保守の問題等はあるのですが、価格面で非常にメリットがあるので、サーバー向きには良い選択かもしれません。

高速化のためのＴＩＰＳ

高速化のためのTIPSです。

・ とにかくシステムのボトルネックを探す、あるいはシステムをつくるときはボトルネックはつくらないように何とかする事です。

重要なのはあるアプリケーションを動かすと、そのアプリケーションはどういう挙動をして何をするか、これを理解していないと、思わぬところにボトルネックができてしまいます。理解していたはずでも、ちょっと失敗しますとボトルネックになるので、できる限りボトルネックはつくらないようにする。

8-2の法則という言葉がありまして、10割の仕事があったとしても、そのうちの２割が本当にボトルネックになる、時間のかかる部分です。その２割の部分を改善すれば、全体としてはスピードアップを計る事ができるという事です。実際にその通りです。

面白いのはインターネットサーバーではCPUよりはディスクアクセスにボトルネックが起きます。

ですから、先ほどのJANOGサーバーが良い例で、CPUはDX4で極めて非力で、インターネットによる接続も普通のISAのネットワークカードでED0というドライバを使って10Mでつながっています。

エンドユーザーさんの回線とは普通64Kや128Kですから10Mといったら確かにすごいわけです。sh.janogの場合ですと、メールを配る瞬間はその10M ISAをフルに使って、大体数秒間それが続きます。その間CPUのアイドルは全くなくなりますが、メールが３分程度で配れれば、その間どんなにCPUが忙しかろうがそれで良いのです。

smtpfeedの場合はディスクアクセスがあまり起きないのでボトルネックにはなりません。

ボトルネックというテーマはなかなか難しいテーマです。

・カタログ性能だけではなく、実際に使ってみて速いディスクを選ぶ事です。

私の経験からすると、カタログスペックと実際のスピードは別のようです。カタログスペックではすごく速そうだけれども、使ってみるとどうも速くないというのがあります。

ハードディスクのアクセスを減らすというのが高速化のためには良いことです。一つはメモリを十分に積んでバッファを効かせる。極端な場合はasync optionを使ってみるという方法も必要だと思います。又、１台当たりのシークを減らすという方法で複数台のハードディスクを利用する方法もあります。

例えば、Webのコンテンツとログは同じディスクには置かないという話です。同じディスクに置くよりは違うディスクにしておいたほうがヘッドの動きが減ります。容量ぎりぎりまで使わない事もあります。ufsの特性ですが、空き容量が２割のディスクと、空き容量が３割のディスクを比較しますと、私の計測した限りではスピードが違います。ですから、ディスクは余裕を持って使う。

もっと極端な使い方になりますが、パーティションを切って使う方法もあります。

例えば、８Gのディスクを持ってきて２Gぐらいしか使わなければ、シークの量は４分の

１ぐらいに経るのではないかという計算ができます。 そこで、２Gでパーティションを切ってしまって使うという方法です。そうすると、ヘッドのシーク量は物理的に制限できますのでかなり速くなります。その他、ハードディスクのスピードを上げるためには転送レートを上げる。
コントローラは複数台用意してCPUの負荷を減らすという方法もあります。

・ もう一つあります。私はこれが好きなのですが人間を鍛える事です。
ハードウエアだけ頑張っても、こういうノウハウは経験値が重要です。私はふだん遅いマシンでテストします。速いマシンだと見えないものが遅いマシンだと気がつく事があるのです。例えば速いマシンだったら0.1秒以下の差になってしまって全然見えないところが、遅いマシンを使ってみると10秒の差になったりします。本当に必要なときは速いマシンを買えばいいのですが、
ふだんは遅いマシンでいろいろテストしてみるというのが一つの例です。
他には、実際には難しいのですが、条件の違うマシンをいろいろ用意して同じことをやってみる。同じことというのは、例えばちょっと大き目のプログラムをコンパイルして時間を計ったり、色々なベンチマークを走らせてみたりするのです。結局、結果に違いが出るわけですから、その違いがどこからくるかを見きわめます。それを今度は速いマシンでちゃんとやれば、より速くなる。
ノウハウというのは聞いただけではなかなか身につかないものですが、実際に痛い目に遭うとよく身につくと思っています。

システムをつくるときのポイント
システムをつくるときのポイントですが、目的をよく見きわめないといけないという事です。
何をするためのサーバーなのかを見きわめる事です。
WWWサーバーなのか、NEWSサーバーなのか、DNSなのか。 スピードも重要ですが、コストも重要で、無制限にお金をかけるわけにはいきません。電力もあります。あまり電気を食わないほうがいいわけです。
それと場所。これも重要なパラメータで、学校のような場所だと比較的余裕はあると思うのですが、例えばプロバイダさんのハウジングサービスを借りるときですと、やはり小さく作っておかなければ同じ場所にたくさん置けません。私はよく「置けなくなったら負けである」という言葉を使っています。要するに、あるスペースがあったときにそこに１台しか置けなくていいのか、10台置けるのかというので違ってきます。同じ能力だったら体積は小さいほうがいいですし、電力も少ないほうがいいです。あとは予算です。これはやはり無視できませんが、予算について言う人は少ない気がします。

まとめ
今まで説明したことはそんなに特別なことではないつもりですが、結局チューニングというのは当たり前のことを全部やればちゃんと速くなるのです。ですから、当たり前のことの積み重ねですので、どこが悪いのかというのを細かくチェックして、それを改善する努力をします。
というところでまとめさせていただきます。

６．質疑応答

Q-1:サーバーを置く位置の問題があるとおっしゃったのですが、その辺について少し教えていただけたら.....。
A：ネットワークロケーションの事です。例えば、エンドユーザーが個人でOCNを引いたとしましょう。 この場合、回線スピードがOCNの一番下のクラスですと128Kですね。これに対してどこに置くかというのは、128Kを満たすだけのサーバーを用意すればいいわけです。これに100万円も1,000万円もかかるシステムを用意する必要はなく、結局はネットワークロケーションです。先ほどのJANOGサーバーの例ですと、10Mでつなぐというのは最初からわかっていましたので、10Mをカバーできるだけの能力があればいい。私の

経験だと、DX2/66でもカバーできてしまうと判断するのです。
ネットワークロケーションをいい場所に置くと当然速くはなるのですが、自分のサーバーだけが速くなっても、結局アクセスする人がPPP主体だったら、同時にたくさん受けられるようにはなりますけれども、あまり速くならない。転送してくる量はあまり変わらないのです。そういう意味で説明しました。もちろん自分の手元に置くよりは、ISPさんやハウジングサービスを借りたほうが、自分のところの回線に負担はかけませんので便利ですが、今度はリモートでのメンテがしやすい、しにくいという問題が起きてきます。コンソールが手元にないと信用できない人はISPのハウジングサービスは不向きです。私の場合、sh.janogというマシンですとDEC PCですので、もし最悪の場合はBIOS画面が出てきますので飛んでいかなければならないのですが、ふだんは完全なリモートでメンテナンスしています。

Q－2：メーリングリストのソフト自体の選択で何かノウハウはありますか？FMＬはいかがでしょうか？他のメールはどうでしょう？
A：私見ですが、FMＬは技術的、アイデア的にはすばらしいのですが、今のインプリメントではだめです。何が問題になるかというと、先ほど説明したメモリにおけるプロセスの存在時間の問題です。FMＬの構造は、SMTPでメールサーバーへ接続して、あて先数(例えば1,000人)分のアドレスをそのsendmailに投げる(指定する)のです。レセプト等でどんどん投げてくるわけです。そうすると、sendmailはその受けたアドレスについていちいちパースを行います。これがすごく重たいのです。アドレスを投げ終わらないとsendmailは次の処理に行かないし、FMLも終われないわけです。FMLはPerlで書いてありますが、その間、Perlの部分とFMLの動く部分のメモリがすごく圧迫されています。今私はメモリ16MBのマシンを使用していますので、これがネックになっています。
問題点はFMLのそこの部分だけです。多分アドレスを１個投げるような形にしてsendmailでincludeの形で展開するようにすれば、その点は解決するのではないかと思います。 FMLの非常にすばらしい点は、例えばduplicateのメッセージの検出や、postの制限等がいろいろできる事だと思います。

Q－3：サーバーユースの場合ですと、サービスをとめないという意味で冗長性を考えることがあると思うのですが、その辺りのTIPSが何かあれば教えて下さい。
A：実は冗長性を考えてもすごく難しいのです。本当に冗長にしたつもりが実は冗長になっていない場合があり、よく考えなくてはなりません。
ご存じのとおり、一番壊れるのがハードディスクです。このハードディスクを保護するための方法として考えられるのがレイドです。レイドシステムといっても、レイドは０、１、２、３、４、５とありますが、冗長性で実用的なものというと、レイド１(ミラー)と、レイド５です。評価したことのある方はご存じだと思いますが、レイド５というのは原理的には絶対に速くならないので、書き込みが大量に発生するところに使うと、それがボトルネックになってしまいます。
ウルトラエンタープライズ450にディスクをつけて４台でレイドしたのと、２台でミラーしたのを比較したのですが、書き込み能力は10倍ぐらい遅くなりました。
レイドする場合ですと、メモリは普通より多めにしてバッファリング効果が高くなるような構成にしなければなりません。壊れそうなものといいますと、後は電源です。サービスを止めないためには、電源は二重化しているシステムを使うしかないです。
例えば、ウルトラエンタープライズ450です。あれはフルコースで載せると電源が２台必要で、３個載せておくと１個の電源が壊れてもきちんと動き続けます。単純なWWWサーバーですと最近Ciscoのローカルダイレクタという製品があります。これを使うと、１台止まっても勝手に切りかえてくれますので、外から見た場合のサービスは止まりません。

ただ、これもユーザーさんのメールサーバーには使いづらいです。データが片方にしかないという状態は困るので、ローカルダイレクタはWWWサーバー向きです。

以上