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

このチュートリアル

- 大規模なマシン
  - 例えば、ISPのNOCで運用するサーバー
- 小規模なマシン
  - 例えば、OCNエコノミーで運用するサーバー
- 適正なシステムを適正なコストで運用する判断が行えればよい
- AT互換機とSunのWSを例に

規模とコスト

- 小規模なシステム
  - システムが安上がり
  - 手間もあまりかからない
- 大規模なシステム
  - お金がかかる
  - 運用にさまざまなノウハウが必要になる
- ラージシステムをスモールコストで運用できれば理想的

速いことは良いことだ

- CPUが速い
  - 計算処理が短時間
- バスが速い
- HDが速い
  - データの読み書きが短時間
  - アプリケーションの起動が短時間
- ネットワークが速い(太い)

スピードとコスト

- イニシャルコスト
  - 概して、速いものは高く、遅いものは安い
- ランニングコスト
  - 速いシステム
    - 手間がかからないので人件費が安くなる(かも?)
  - 遅いシステム
    - 手間がかかるので人件費はかえって高くなる(?)

インタネットサーバーの例

- 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キャッシュサーバー
- IRCサーバー

システム構成

システム構成のポイント

- CPUの種類とスピード
  - PC-ATを例にすると
    486マシンからPentium-III XeonやAthlonまで
- メモリ容量
  - 16M程度から512M、あるいはそれ以上
- ハードディスク
  - どの容量のディスクを何台？
- Network I/F
  - 10Mbps or 100Mbps

CPU選択のポイント

- アーキテクチャ
  - x86, SPARC, PowerPC, Alpha….
- クロック
- キャッシュ
- メモリ

CPUアーキテクチャ

x86	PC-AT互換機
SPARC	Sunとその互換機
PowerPC	Macintosh, IBM?
MIPS	SGI
Alpha	COMPAQ
PA-RISC	HP, HITACHI

AT互換機のCPU(1/2)

- 過去(?)のCPU
  - i486　25～100MHz
  - Pentium　75～200MHz
  - Pentium MMX　166～266MHz
  - Pentium-Pro　150～200MHz
- SPECint95で0.3～8

AT互換機のCPU(2/2)

- 現行のCPU
  - Pentium-II　233～450MHz
  - Pentium-II Xeon　400～450MHz
  - Pentium-III　450～600MHz
  - AMD K6-III　350～
  - AMD Athlon　500～700MHz
- SPECint95で9.5～22

SPARC(1/2)

- すでに過去(?)のCPU
  - MicroSPARC(LX, Classic) 50MHz
  - MicroSPARC-II(SS5)70～110MHz
  - TurboSPARC (SS5)　　170MHz
  - SuperSPARC(SS20等)
  - HyperSPARC

SPARC(2/2)

- 現行CPU
  - UltraSPARC (Ultra1)　　167MHz
  - UltraSPARC IIi(Ultra5,10) 270～480MHz
  - UltraSPARC II (EP450他) 250～450MHz
- SPECint'95 で 6.6～20程度

CPU Clock Speed

- 最も単純な指標
- クロックスピードと処理能力は密接な関係
- 200MHzのCPU vs 400MHzのCPU
  - 倍速くなるとは限らない
  - 倍以上に速くなることもある

CPUの能力の違い

- FreeBSDのカーネルコンパイル
  トータル時間での比較
  - Celeron 466　　2分以下
  - Pentium-II/333　　2分程度
  - Pentium/120　　10分程度
  - 486DX2/66　　20〜40分程度
    - 注: FreeBSDのバージョンやシステム構成が違うため単純に比較できない。

Cache Memory(1)

- CPUとメモリのバス速度の差を吸収する
- 1次キャッシュ
  - CPUに内蔵されている
  - 4K ～ 64Kbyte
- 2次キャッシュ
  - 多くはCPUの外部
  - 128k～2Mbyte
    - 2次キャッシュが0のシステムもある

Cache Memory(2)

- サイズ
  - キャッシュメモリそのものの量
- キャッシュバスのスピード
  - CPUクロックとアクセススピードの関係？
- キャッシュ可能エリア
  - 主記憶サイズとの兼ね合い

キャッシュサイズ(x86)

- Pentium-Pro
  - 1次8k/8k 2次256k～1M
- Pentium-III
  - 450MHz超　1次 16k/16k　2次 512K
- AMD K6
  - 1次 32k/32k　2次はMBの構成による

キャッシュサイズ(SPARC)

- UltraSPARC-IIi (Ultra5/Ultra10等)
  - 1次　16K/16K
  - 2次　270MHz　256K
    300MHz　512K
    333MHz　2M
- UltraSPARC-II
  - 1次　16k/16k
  - 2次　250MHz 1MB　300MHz 2MB
    400MHz には 8MBのものもある

Pentium-II vs Pentium-Pro

- Pentium-II 233～333MHz(66MHz)
  - Cache Clock = 1/2 CPU Clock, Area ≦ 512KB
- Pentium-Pro ～200MHz
  - Cache Clock = CPU Clock, Area ≦4GB
- Cacheにヒットする限り P6-200は速い
  - 計算ならPentium-II
  - 大規模なデータを扱う場合Pentium-Pro

Celeron vs Pentium-III

	Celeron	Pentium-III	Pentium-III(E)
1次Cache	16K + 16K	16K + 16K	16K + 16K
2次Cache	128K	512K	256K
Cache clock	CPU	½ CPU	CPU
FSB	66M	100 or 133M	100 or 133M
CPU Clock	～500MHz	～600MHz	～733MHz
お値段	安い	高い	かなり高い

メモリバス

- CPUとメモリのインターフェース
- メモリバスのクロックとバス幅で決まる

バス幅	クロック	転送量
32bit	33MHz	132MByte/sec
64bit	66MHz	528MByte/sec
64bit	100MHz	800MByte/sec
128bit	100MHz	1600MByte/sec

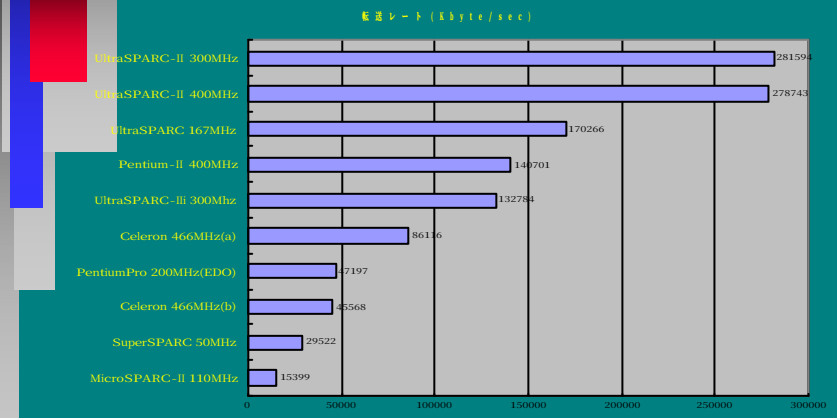
実際のメモリの挙動

- 計算通りの転送レートは得られない
  - 最初のアクセスに時間がかかる
  - 1クロックで転送できるとは限らない
  - 4-2-2-2　最初の1ワードに4クロック
残りの3ワードは6クロック
  - 5-1-1-1　最初の1ワードに5クロック
残りの3ワードは3クロック

実際のメモリ転送速度の差(1/2)

- memcpy によるメモリ転送速度を比較
  - for (i = 0 ; i<= n; i++) memcpy(dst,src,len); にかかる時間をgettimeofdayで計測
- 10M を 100 回(トータル1G)程度での計測
  - ある程度時間がかかる大きさを選ぶ
    - 時間計測のゆらぎを減らす

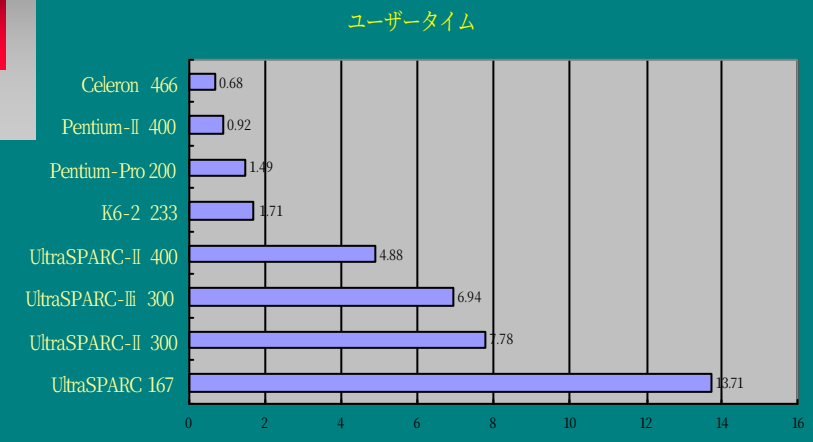
実際のメモリ転送速度の差(2/2)



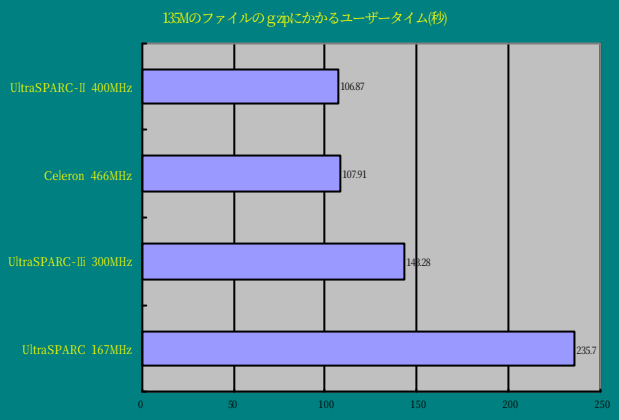
メモリ容量

- 必要にして十分なメモリを用意する
  - 少ないとパフォーマンスに影響する
  - 多すぎるメモリは単なる無駄
    - WS用のメモリは安いとはいいがたい
- 次のことを頭にいれる
  - 稼動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

SPARC vs x86
ssh-keygenの時間の目安



SPARC vs x86
gzipにかかる時間(目安)



SPARC vs x86

- 概して、整数演算はx86が速い
- 概して、浮動小数点演算はSPARCが速い
- 概して、データ転送はSPARCが速い
- 実際は条件によって変わる

ハードディスク

- 磁気円盤が高速回転し
  磁気ヘッドで読み書きを行う
- 連続したアクセスは速い
- ランダムなアクセスはヘッドが動くため遅い
  - **インターネットサーバーのボトルネックの要因**
- サーバー用にはSCSIが一般的
  - IDEも実用上問題無い

実際のHDの例

- IBM DDRS シリーズのカタログより
  - DDRS-34560 and DDRS-39130
- スペック
  - 109-171 Mbits/sec Media data rate
  - Rotational speed 7200 rpm
  - Sustained data rate 8.3-13.3 MB/s
  - Average seek time 7.5 ms
  - Average latency 4.33 ms

SCSIコントローラ

- DMA方式
  - コマンドを送ればコントローラが転送
  - CPUは次の作業へ進める
- WRITE　コマンド送れば完了
- READ　結果を待つ必要あり
- 複数のコントローラ
  - 同時書き込みが可能になる

Network I/F

- あまり選択の余地がない
- Ethernet
  - 10M or 100M
  - 良いSwitchと組み合わせてfull-duplex
- FDDI
  - 高価なのがネック
- 最近のCPUは100Mbpsを使いきる能力

Operating System

OSのチューニング

- 与えられたまま使っていては負け
  - 不要なdaemon類は動かないようにする
    - Solarisは多くのdaemon processが動く
    - FreeBSDの場合、不要なものはあまり動かない
  - 変更できるカーネルなら余計なドライバを削る
    - BSD, Linux等…
- メモリの節約、プロセステーブルの節約
  - システム全体のスループットの向上

Solaris 2.6の標準状態(1/2)

- ps -efaの結果

```
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0   Oct 13 ?       0:01 sched
    root     1     0  0   Oct 13 ?       0:00 /etc/init -
    root     2     0  0   Oct 13 ?       0:00 pageout
    root     3     0  0   Oct 13 ?      13:12 fsflush
    root   195     1  0   Oct 13 ?       0:00 /usr/lib/sendmail -bd -q1h
    root   167     1  0   Oct 13 ?       0:03 /usr/sbin/cron
    root   277     1  0   Oct 13 ?       0:00 /usr/lib/saf/sac -t 300
    root    75     1  0   Oct 13 ?       0:00 /usr/sbin/aspppd -d 1
    root    96     1  0   Oct 13 ?       0:00 /usr/sbin/in.rdisc -s
    root   106     1  0   Oct 13 ?       0:00 /usr/sbin/rpcbind
    root   133     1  0   Oct 13 ?       0:00 /usr/sbin/inetd -s
    root   108     1  0   Oct 13 ?       0:00 /usr/sbin/keyserv
    root   153     1  0   Oct 13 ?       0:00 /usr/sbin/syslogd -n -z 12
    root   138     1  0   Oct 13 ?       0:00 /usr/lib/nfs/statd
    root   140     1  0   Oct 13 ?       0:00 /usr/lib/nfs/lockd
    root   173     1  0   Oct 13 ?       0:00 /usr/sbin/nscd
    root   183     1  0   Oct 13 ?       0:00 /usr/lib/lpsched
    root   280     1  0   Oct 13 ?       0:00 /usr/dt/bin/dtlogin -daemo
```


Solaris 2.6の標準状態(2/2)

```
    root   212     1  0   Oct 13 ?       0:02 /usr/sbin/vold
    root   218   216  0   Oct 13 ?       0:00 /usr/sbin/ccv -f
    root   205     1  0   Oct 13 ?       0:00 /usr/lib/utmpd
    root   216     1  0   Oct 13 ?       0:00 /usr/sbin/cssd
    root   217   216  0   Oct 13 ?       0:00 /usr/sbin/cs00
    root   219   216  0   Oct 13 ?       0:00 /usr/sbin/kkcv -f
    root   221     1  0   Oct 13 ?       0:00 /usr/lib/locale/ja/wnn/dpkeyserv
    root   225     1  0   Oct 13 ?       0:00 /usr/lib/locale/ja/wnn/jserver
    root   226   225  0   Oct 13 ?       0:01 /usr/lib/locale/ja/wnn/jserver_m
    root   281   277  0   Oct 13 ?       0:00 /usr/lib/saf/ttymon
    root   278     1  0   Oct 13 console  0:00 /usr/lib/saf/ttymon -g -h -p xxxxx console login:  -T
sun -d /
    root   260     1  0   Oct 13 ?       0:00 /usr/lib/snmp/snmpdx -y -c /etc/snmp/conf
    root   270     1  0   Oct 13 ?       0:00 /usr/lib/dmi/snmpXdmid -s xxxxxx
    root   282   260  0   Oct 13 ?       0:00 mibiisa -p 32788
    root   269     1  0   Oct 13 ?       0:00 /usr/lib/dmi/dmispd
    root 11379     1  0   Nov 06 ?       0:00 /usr/openwin/bin/fbconsole -d :0
    root 11376   280  0   Nov 06 ?       0:02 /usr/openwin/bin/Xsun :0 -nobanner -auth /var/dt/A:0-
8Iv0z_
    root 11301   133  0   Nov 06 ?       0:00 /usr/dt/bin/rpc.ttdbserverd
    root 11393 11377  0   Nov 06 ?       0:01 dtgreet -display :0
    root 11377   280  0   Nov 06 ?       0:00 /usr/dt/bin/dtlogin -daemon
```

停止したサービス

- インターネットサーバーに OpenWindowsは不要
  - 必要ならば手動で起動することにして、セッションマネージャ類を止める
- 仮名漢字変換関連のdaemonを止める
  - Sol 2.6の場合、適当にインストールするとWnnとATOKのサーバーが動く

停止したサービス(2)

- NFS, RPC, NISは利用しない
  - inetd.confからrpc関連をコメントアウト
  - rpcbind(いわゆるportmap)を停止
- その他
  - vold, snmp, lpsched等
- 場合によっては sendmailも止める
  - sendmail -q30m として-bd を抜く方法もある(セキュリティ向上のため)

Solaris 2.6のチューニング後

- ps -efaの結果

```
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0   Sep 26 ?       0:00 sched
    root     1     0  0   Sep 26 ?       0:18 /etc/init -
    root     2     0  0   Sep 26 ?       0:00 pageout
    root     3     0  1   Sep 26 ?     390:47 fsflush
    root   146     1  0   Sep 26 ?       0:00 /usr/lib/sendmail -bd -q1h
    root   214     1  0   Sep 26 ?       0:00 /usr/lib/saf/sac -t 300
    root   104     1  0   Sep 26 ?       0:18 /usr/sbin/inetd -s
    root    94     1  0   Sep 26 ?       1:10 /usr/sbin/in.named
    root   109     1  0   Sep 26 ?       0:21 /usr/sbin/syslogd -n -z 12
    root   126     1  0   Sep 26 ?       1:02 /usr/lib/inet/xntpd
    root   136     1  0   Sep 26 ?      13:51 /usr/sbin/nscd
    root   130     1  0   Sep 26 ?       0:16 /usr/sbin/cron
    root   156     1  0   Sep 26 ?       0:01 /usr/lib/utmpd
    root 19855     1  0   Oct 30 console  0:00 /usr/lib/saf/ttymon -g -h -p xxxx console login:
-T sun -d /d
    root   221   214  0   Sep 26 ?       0:00 /usr/lib/saf/ttymon
```

- Sol 2.xは32Mmem程度でも実用になる
  - できることは限られるので注意

Solarisのインストール(1)

- Solarisのトータルメモリエリア
  - 実メモリ＋HD上のスワップサイズ
- スワップサイズは0に
  - 基本はスワップしたら負け
  - 64M程度では、CDE利用できず
  - 最悪の場合 mkfile と swap -a
- 必用なだけメモリを搭載する
  - 純正メモリでなければ256MBで10万円程度

Solarisのインストール(2)

- Coreインストール
  - インターネットサーバーに
    デスクトップ環境は不要
  - Solaris ではなく SunOS 5.x ?
- 不足するパッケージは後から追加
  - ディスク消費量は100M以下で収まる
    - フルインストールだと1G近く消費
    - インストール時間も極めて短くなる

Solarisのインストール(3)
不足パッケージの検索

- コンパイルしてみると判明
  - stdio.h とかが無い
- OSのメディアから

```
% fgrep stdio.h */pkgmap
SUNWhea/pkgmap:1 f none usr/include/stdio.h
                 0644 bin bin 15398 47206 869024258
```

Solarisのインストール(4)
追加パッケージ

- コンパイル環境を用意する

system	SUNWsprot	Solaris Bundled tools
system	SUNWtoo	Programming Tools
system	SUNWbtool	CCS tools bundled with SunOS
system	SUNWhea	SunOS Header Files
system	SUNWarc	Archive Libraries
system	SUNWntpr	NTP, (Root)
system	SUNWntpu	NTP, (Usr)
system	SUNWscpu	Source Compatibility, (Usr)
system	SUNWlibC	SPARCompilers Bundled libC
system	SUNWlibm	Sun WorkShop Bundled libm

サービスの止めすぎに注意

- **不要だと思って止めたところで…..**
  - **OpenWindowsを起動したら、起動するまでに異様な時間がかかる**
  - **Netscape Enterprise Serverが起動しない**
    - **apacheは動くのに …**
  - **あたりをつけて、1つづつサービスを動かして確認する(しかない)**
- **止めるとパフォーマンスを落とすサービス**

FreeBSDのGENERICカーネル

- インストールがだいたいうまくいくような、最大公約数的カーネル
  - SCSIドライバが10種類以上
  - ネットワークドライバも10種類以上
  - CD-ROM用ドライバがいろいろ
- カーネル内部のテーブルも小さい
  - 1人のユーザで使う分には、ほぼ間に合う
- とりあえず使うにはなんとかなる程度

リコンフィグ

- 不要なファイルシステム、ドライバを削る
  - NFS, MSDOSFS, CD9660… ed0, aic0, nca0…
- テーブルを必要な大きさに増やす
  - maxusers, NMBCLUSTERS, FD_SETSIZE…
- LINT(/sys/i386/conf)を眺めながらGENERICを修正する
- その他、各種パラメータのチューニング

FreeBSDのカーネル

- インストール直後はGENERICカーネル
- カーネルのリコンフィグを行う

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
```


Solarisの場合

- /etc/systemのチューニング
  - FDやpty, 共有メモリなどの変更
  - AnswerBook, Solaris FAQを参考に…
    - 変更後はリブートが必要
- nddコマンドと/dev/tcpによる tcpパラメータのチューニング
  - /usr/sbin/ndd /dev/tcp ?
  - 変更が必要になるのは、特別な場合

Server Software

WWWサーバー

- HTTPのリクエストを処理するサーバー
  - インターネットでもっとも稼動台数が多い?
- HTTP処理
  1. TCP接続
  2. クライアントからのリクエスト
  3. サーバーのレスポンス
  4. TCP切断
  - TCP Connectionの断続

WWWサーバー

レスポンスまでの処理

- リクエストの解析
  - どのページ?、CGI?、URLのmapping?
- ページデータの読み出しやCGIの実行
- ログの生成
  - (必要なら)IPアドレスからFQDNへのネームサーバーへの問い合わせ
- レスポンス

WWWサーバー

ボトルネックの要因(1)

- コンテンツデータの読み出し
  - Disk I/O
- cgiプログラムの実行
  - CPUパフォーマンス
  - メモリ不足によるスワップ
  - プログラムの実行が遅い
    - C vs Perl ?

WWWサーバー

ボトルネックの要因(2)

- ログの書き出し
  - 100万hit/dayの場合、10行/秒を越える
    - ヘビーなサーバーではログファイルも無視できない
  - DNSを調べてFQDNで記録する場合の問題
    - ログの記録が遅くなる
- ネットワークロケーション
  - サーバー側はISPのハウジングサービス

WWWサーバー

1requestの負荷

- リクエスト数×処理時間 を恒に考慮する
- 1リクエストに1秒かかり、メモリが1M
  - 10リクエスト/秒なら、メモリは10M
  - 100リクエスト/秒なら、メモリは100M
- CPUが10倍速なら、処理時間は0.1秒
  - 100リクエスト/秒でも、メモリは10Mですむ
- ネットワークの転送スピードも考慮
  - クライアントが遅いと転送時間は変わらない

WWW Server Software

- 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を利用
- thttpd　select を使ったループで処理

CERN Server

- WEB Serverのサンプルインプリメント
- WEB Proxyサーバーとキャッシュ
  - 兼用サーバーが簡単に…
- 1リクエスト毎にfork
  - リクエストを受けた後にfork
  - UNIX向きの処理方法だが、
    forkの処理は重くパフォーマンスの低下
- 最近はあまり使われてない?

NCSA httpd

- 高機能のWEBサーバー
  - URLのリダイレクト
  - アクセス制限
  - 柔軟な設定
  - CERNよりは高速
- 機能が高いため一時広く利用される

Apache WEB サーバー

- 正確な動作を重視したインプリメント
  - スピードよりも正確さを優先
  - しかしながら十分高速
- 予めforkしたプロセスを用意
  - リクエストに応じて子プロセスを順に利用
- バーチャルホスト対応
- 現在世界でもっとも稼動台数の多いWEB

phttpd & thttpd

- phttpd
  - thread を利用し、速いらしい。
    - Netscape Enterprise Serverも同様の構造
  - 移植性に難あり
    - ほとんど Solaris 専用
- thttpd
  - 複数コネクションを selectを使って同時に処理
  - かなり高速だが、基本機能のみ
  - セキュア

CGIのおさらい

- Common Gateway Interface
- WEBサーバーから外部プログラムを実行
  動的にページイメージを作成し
  その結果を表示
- 身近な例
  - Web BBS では、Perl 等と組み合わせて
    広く使われてる

CGIの処理の流れ(1/2)

- 外部プロセスの生成
  - fork→exec
  - UNIX のプロセス生成の基本処理
- fork
  - 一つのプロセスが二つになる
  - UNIX での重い処理
    - データ領域のコピーが発生する
    - copy on write があれば…

CGI処理の流れ(2/2)

- exec
  - 実際のプログラムの実行
- fork → exec には
  多少なりとも時間がかかる
  - copy on write がなければ
    データサイズが大きいほど遅い
    - copy on write があっても….

実サーバーの例

- Netscape Enterprise Server
  - Thered ベースのサーバー
    - html の処理 → thread
    - CGIの処理 → fork
  - Theredは軽く高速
- Apache
  - forkベースですべて処理
    - html処理 → fork
    - cgi処理 → fork
  - fork部分の処理を予め高速化

両者の比較

- Netscape Enterprise Server
  - 何もしなくても10M程度のプロセス
    - リクエストが増えるにつれてメモリ増加
      - 800k 程度？
    - 大きくなると50Mを超えるプロセス
- Apache
  - 一つ一つは 2M〜3M程度のプロセス
  - リクエストに応じてどんどん増える

CGIでの差

- fork からexecまでのわずかの時間に
  大量のリクエストが溜まると
- Netscape Enteprise Server
  - 50Mのプロセスがforkしたら怖い
  - 瞬時にメモリ消費
    - 20リクエストで1Gのメモリを消費
- Apache
    - 3M程度のプロセスが増えるだけ
    - 20リクエストでも60M

NEWSサーバー

- トラフィック 90万通/day 18〜20GByte/day
  - 現状もっともヘビーなサーバー
- 他のサーバーとの記事の交換
- エンドユーザー向けのサービス
  - 記事の購読、投稿
- コントロール処理

NEWSサーバー
記事の受信

- 他のサーバーからの受信
  - 記事の重複の検索
  - スプールへの書き込み
  - 送信するためのファイルの生成
    - いずれもDisk I/Oの発生
- full feedのボリュームを受けるのは大変
  - T1では足りない

NEWSサーバー
記事の送信

- 他のサーバーへの記事の送り込み
  - 記事の送信ファイルの読み込み
  - スプールから読み取った内容の送信
    - 配送先が増えるにつれて、ディスクI/Oの増大

NEWSサーバー
ユーザーの購読用

- 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対応
  - 対象とするユーザー数
- cancel処理

従来のINN (before 1.x)

- スプール形式にufs (UNIX File System)
- 記事番号のファイル名
- ニュースグループのディレクトリ
- news.software.nntpの350番
  - /var/spool/news/news/software/nntp/350
- cancelが重い
  - ufsがボトルネック

ufsの問題

- ファイルを作る
  - 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
- ファイルを消す
  - 目的のファイルが見つかるまでリニアサーチ
- ディレクトリに大量のファイルがある場合のパフォーマンス低下を招く

Diablo

- 元々配送専用サーバー
  - 当初ユーザーが読む機構は無かった
- キャンセル処理をしない
- activeやnewsgroupsファイルが無い
  - 購読用ならある
- INN 1.xの5〜10倍のパフォーマンス
- news spoolは独自の形式

現在のINN (version 2.x)

- CNFS (Cyclic News File System)
- 巨大なファイル内部に記事を格納
- 先頭から記事を置き、
  最後まで使ったらまた先頭へ
- history には記事のファイル中の
  offsetを記録
- cancelが劇的改善

Mailサーバー

- 2種類のMailサーバー
  - メーリングリスト用メールサーバー
  - PCユーザーなどのPOP&SMTPサーバー
- それぞれ、チューニングポイントが違う
  - より速く大量に
  - より多くのユーザーを1台のマシンで

メーリングリストサーバー

- 多くの宛先にメールを送る
  - できる限り高速に
- 配送が遅いと議論がかみ合わない
  - sendmailは1アドレスづつ順に配る
- qmail
  - 小さなプログラムで次々配送
- sendmail + WIDE patch + smtpfeed
  - selectを利用し同時に複数の宛先へ配送

sendmailの特徴

- 逐次配送
  - 前の配送が終わらないと次を配送しない
    - 配送先がたまたまアンリーチだったりすると、その後の宛先全部が遅れる
- 同一MX先の相乗り
  - user1@foo.co.jp user2@sh.foo.co.jpが同じMXなら1通で配送

qmailの特徴

- 複数の小さなプログラムを組み合わせ用途別に処理
  - qmail-smtpd, qmail-inject, qmail-remote…
- security holeを発生しないような構造
  - security holeの発見に$1,000の賞金
- 軽くて高速
- 同一ドメインであっても1通づつ配送

sendmail + WIDE Patch + smtpfeed

- smtpfeedが実際の配送を行う
  - sendmailの外部メーラ
  - sendmailとはLMTPで通信
    - WIDE patch が必要
- MXの相乗り
- select を使って複数に同時配送
  - きわめて高速
  - DX4で 1600アドレスを3分以内に90%配送

POP,SMTPサーバー

- SMTPについてはメールサーバーと同様
  - SPAM対策はちゃんとする(おまけ)
- qpopper
  - もっとも普及しているpopサーバー
- 同一ディレクトリに大量ファイルの問題
  - ufsボトルネック問題

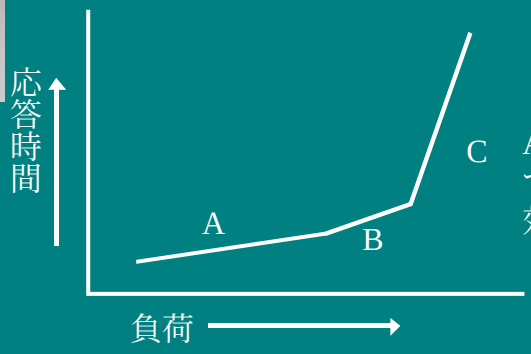
qpopperの注意点

- qpopperの挙動
  - リクエストがあるとメールボックスを1行づつテンポラリファイルにコピー
  - 接続終了時に、元のファイルに書き戻す
- テンポラリファイルを置くディスクを別にするように改造する
  - MFSの利用も考える
    - ちょっとだけ危険だが…

OSのインストール

- 不要なものまでインストールしない
  - Solarisの場合、CDE環境無しでも良い?
- パーティションの切り方
  - 『swapサイズはメモリの倍とる』は過去の話
    - メモリ512Mでswap 1Gなどというのは愚の骨頂
  - 十分なメモリを用意したらswapは0でもよい
    - OSによってはswap 0が不可なものもあるので注意

負荷と応答時間



AとBの境ぐらいで運用するのが効率が良い

運用開始後の監視

- 正常に動作しているか
- ボトルネックはないか？
- vmstat, iostat, netstat, ps などでチェック
  - あるマシンの正常運用状態
  - 破綻状態

さらなる高負荷への対応

ラウンドロビンDNS

- 複数のサーバーを1台に見せるテクニック
- 高負荷を複数サーバーで処理する。
  - 1台のサーバーでは処理が間に合わない
- サービスによってはできない場合もある
  - データベース等
- WEBサーバーで多用される
  - 新聞社や検索エンジンのサーバー
  - 大手企業の人気サーバー

Aレコードによるラウンドロビン

- www IN A 10.10.10.1
  IN A 10.10.10.2
  IN A 10.10.10.3
- Aの問い合わせには順番を入れ替えて返す
  - 1つめの問い合わせ
    10.10.10.1,10.10.10.2,10.10.10.3
  - 2つめの問い合わせ
    10.10.10.2,10.10.10.3,10.10.10.1
- クライアントは最初のAレコードから順に試す

CNAMEによるラウンドロビン

- ゾーンファイルの設定例

```
www     IN    CNAME    www1
        IN    CNAME    www2
        IN    CNAME    www3
www1    IN    A        10.10.10.1
www2    IN    A        10.10.10.2
www3    IN    A        10.10.10.2
```

- 実は禁止されている設定だが動作する
  - bind8 では multiple-cnames　yes; とする

CNAMEラウンドロビンの挙動

- クライアントがキャッシュネームサーバーに問い合わせる
- キャッシュネームサーバーがマスターネームサーバーへ問い合わせる
- マスターネームサーバーはCNAMEの1つを返す
- キャッシュサーバーはその結果をキャッシュ

ラウンドロビンの比較

	A	CNAME
マスターからキャッシュへ	全てのA	1つのCNAME
キャッシュするもの	全てのA	CNAMEとA
クライアントへ返す	毎回違う順のA	同じA
クライアントのアクセスするサーバー	全てのサーバーを順にアクセス	特定のサーバーをアクセス

ラウンドロビンDNS

- Aによるものは全てのサーバーを満遍なくアクセスするようにできる。
- CNAMEによるものは、特定のサーバーをアクセスするようにできる。
- WindowsのWEBブラウザは、複数Aがあっても特定のサーバーをアクセスするものがある。
  - Proxyサーバーを経由するときは Proxyサーバーの挙動による。

Real Example

news.nspixp.wide.ad.jp [1996/10]

- IXでのHUBとなるニュースサーバー
  - WIDE Projectのnspixpの一環
  - 当時のトラフィックを支えてそこそこ速いこと
- スペック
  - P5-133, Mem 128MB
  - HDは 2G × 2 + 4G
    - 2Gはシステムとhistory、4Gはスプール
    - SCSIは2チャンネル

news.nspixp.wide.ad.jp日記 (1)

- ソフトウェアはdiabloを利用
- CPUの余裕が少なくなったためP6-200へ
- Etherを 10Mbps→100Mbps に変更
  - 100Mのつもりが、Etherスイッチの設定ミス
- mount optionにnoatimeを追加
  - ちょっとだけパフォーマンス向上
  - async optionも試してみたが効果なし.
    おそらくメモリ不足（この時点では128M)

news.nspixp.wide.ad.jp日記 (2)

- diablo を shmを使うように修正
  - configのミスで使われていなかった
- incoming 18.57GB/日の記録
  - outgoing 43.17GBと減る
- メモリを128MB → 256MB
  - スプールのexpire時間が半分に短縮

news.nspixp2.wide.ad.jp [1998/1]

- Pentium-II 333
  - スピードではなく入手しやすさと、発熱量
- 384MB メモリ
  - MBの限界　(128MB DIMM x 3)
- スプールのI/Oを稼ぐ
  - 9G×2 を ccd でストライプする
- 100Mbps ネットワーク
  - nspixp2の事情によりFDDI

news.nspixp2.wide.ad.jpの実力

- 31のISPと接続
- トラフィック
  - incoming　55〜80万通　18〜23GB
  - outgoing　800〜 1000万通　270〜330GB
- 最大トラフィック
  - 1998/12/6　in 25.7GB　out 361GB
- ピーク時はFDDI帯域を使い切る

現在のnews.nspixp2.wide.ad.jp

- すでに処理能力が限界を超える
  - ボトルネックはネットワーク帯域
- 次期マシンはGigabit I/Fを利用
- CPUもさらに強化
- メモリは768M
- 現在発注中
  - 年内稼動が目標

sh.janog.gr.jp [1998/8]

- JANOGのメール、WWWサーバー
- ハードウェア
  - CPU DX4ODP100
  - メモリー 16M
  - Hard Disk SCSI 2G × 1
  - MB ISA/VL
- お金をかけずに作るのが目標
  - ベースは、ゴミになる直前を拾ってきた(^^;

sh.janog.gr.jp (2)

- Software構成
  - OS　FreeBSD 2.2.7 RELEASE
  - www　thttpd 2.08
  - mail　sendmail 8.9.3 + WIDE patch 3.2W + smtpfeed 1.02
- メモリ16Mで現状十分
  - 定常的なprocessが増えたりしない
  - ログインしたときのシェルの使うメモリが…

Solaris 2.xでの例

- Ultra1で512Mmemなマシンがあり 1Gのメモリが必要になった
  - 6~70M規模のプロセスが数個常駐
- Sun純正メモリは定価で200万円近い
  - 実売は110万円ぐらい??(1998年秋)
- UltraSPARC 167MHzで メモリを1Gも積んで大丈夫？
  - システムバランスの問題

Solaris 2.xでの例

SPARCengine Ultra AXiの導入

- SUNのOEM用プロダクトである
  SPARCengine Ultra AXiを利用
  - ATXのボードに300MHz UltraSPARC-IIi
  - CPU＋ボード、1Gメモリ、4.3G HD x 2、
    2nd Ether I/F、VIDEO、19inch 4Uなケース
  - モニタ無し、CD-ROMドライブもなし
- PCを組み立てられない人、
  Sunに詳しくない人にはお勧めしない！

高速化のためのTIPS(1)

- システムのボトルネックを探す
- 8-2の法則
  - 8割の処理が2割の部分で行われている
  - その2割の部分を改善すれば、
    全体では大きく性能アップ
- インターネットサーバーでは、多くの場合
  CPUよりディスクアクセスにある
  - 見つけにくい場合も多い

高速化のためのTIPS(2)

- 速いHDを選ぶ
- HDのアクセス(head seek)を減らす
  - メモリを十分に積んでバッファを効かせる
  - 物理的に複数台のディスクを利用し、1台あたりのアクセスを減らす
    - ストライプは効果的
  - 容量ぎりぎりまで使わない
- HDの転送レートを上げる
  - コントローラを複数用意する

高速化のためのTIPS(3)

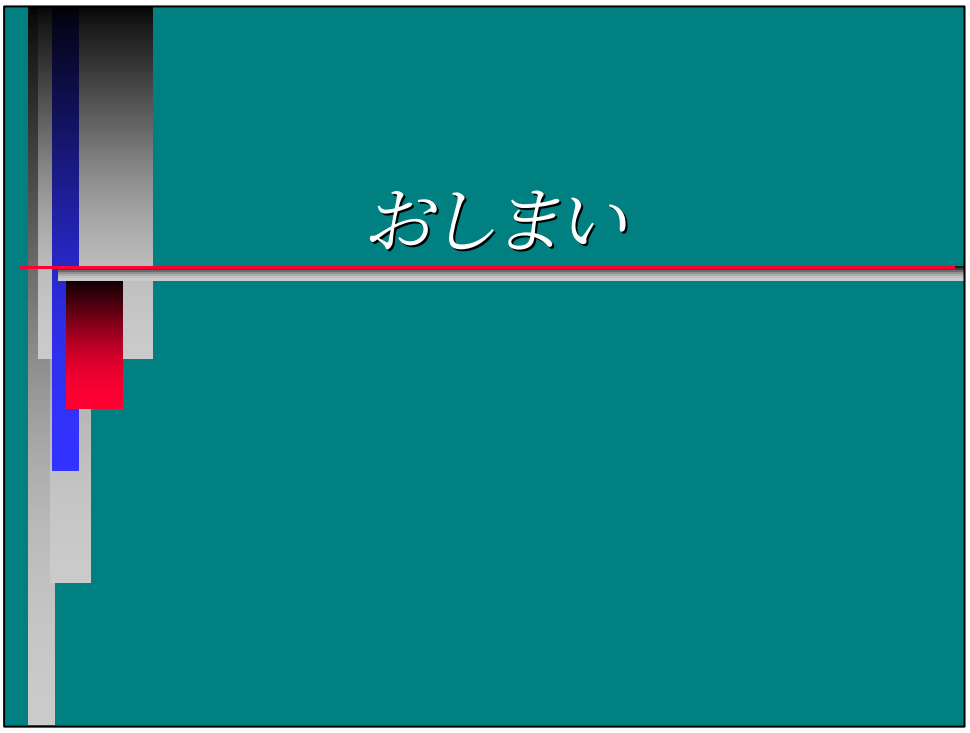
- 人間を鍛える
  - 普段は遅いマシンを使う
  - 速いマシンでは気がつかない差が遅いマシンだと気がつきやすい
  - 条件の違うマシンをいろいろ用意して同じことをやってみる
  - 結果の違いはどこからくるのかを見極める
- ノウハウは聞いただけでは身につきにくい

システムを作るときのポイント

- 目的をよく見極める
  - スピード?、コスト?、省電力?、省スペース?
- 予算は絶対に無視できない
  - 余ってるものがあるなら使ってみる
- 置けなくなったら負け
  - 必要とするスペースと電力も考慮する

まとめ

- 限界のように見えるマシンでも、能力を活用していないことが多い
  - ボトルネックをさがす
  - リソースは無駄なく有効に活用する
- CPUとネットワークは速く、ディスクは遅い
  - バランスの良いシステムを構築する
- チューニングは当たり前のことの積み重ね
  - 一つ一つ細かくチェックする


おしまい