ネットワーク・サーバの パフォーマンス チューニングのための基礎実験

田中二郎*

概 要

ネットワーク・サーバを効率よく稼動させ、パフォーマンスを向上させることは、システム構築・維持においての重要な目標のひとつである。そのために、アプリケーションレベルでのベンチマーク・プログラムや解析ソフトが数多く考案されている。

一方、ＣＰＵ、メモリ、ネットワーク、ディスク・ドライブといった個々のハードウェアの構成要素についても、多くのベンチマーク・プログラムが考案されている。ただ、こうした測定結果は、アプリケーション的な観点からの平均値の測定であって、ハードウェアの限界性能を検証したものは少ない。

本研究では、まず個々のハードウェアの構成要素の限界性能を計測し、その計測値とデータシートより計算される理論値とを比較し、計測方法の妥当性を検証する。次に、個々のハードウェアの構成要素が限界となるような環境でのパフォーマンス測定をおこない、アプリケーション全体のチューニングへの一助とする手法を提案する。

—キーワード—
ネットワーク・サーバ、パフォーマンス・チューニング、ベンチマーク

1 基礎実験

1.1 実験装置

本実験では、サーバを１台、負荷をかけるためのクライアントを３台使用した。

サーバのハードウェアは、ASUS 製のマザーボード P5LD2-V[(1)]を主として自作したものを使用した。P5LS2-V は、LGA775 ソケット CPU に対応し、チップセットはインテル 954G と ICH7R を使用している。CPU には動作周波数 3GHz のインテル Pentium4 を使用した。メモリは 512MB を 2 枚、計 1GB を搭載した。ハードディスクは、日立製の Deskstar 7K500 などを使用した。ネットワークインターフェースは、動作実績の高いインテル PRO/1000[(2)]を PCI バスに挿入し使用した。このネットワークインターフェースは、負荷用のクライアントにも使用した。

*Jiro TANAKA 日本橋学館大学人文経営学部

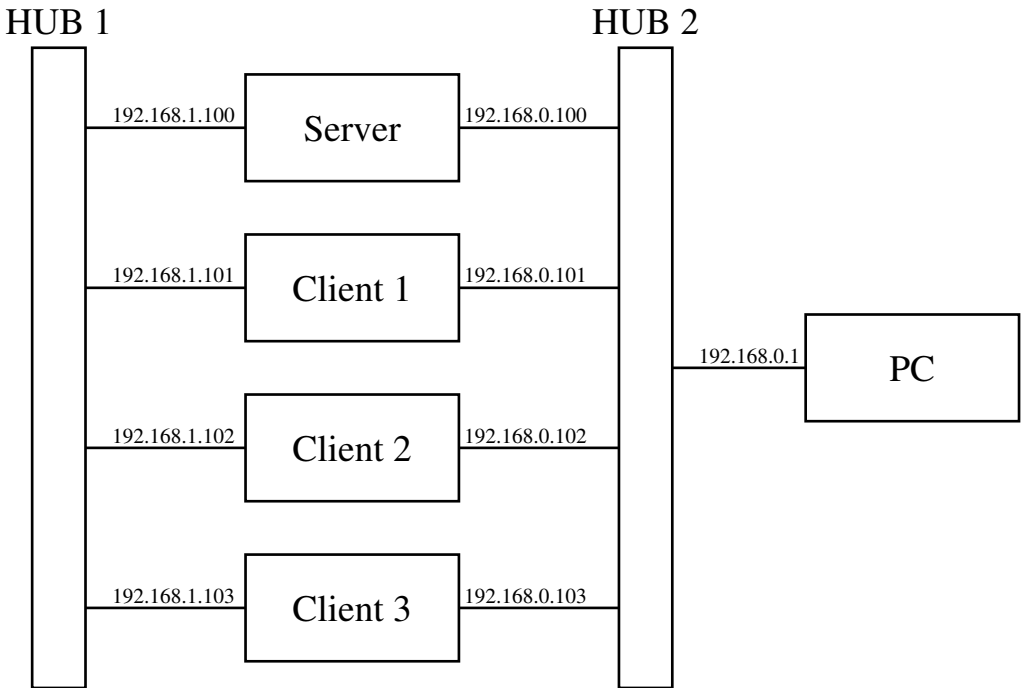

図 1: 接続図

クライアント３台のうち２台は、1.5GHz のインテル Pentium4 を使用し、残りの１台はインテル Celeron 566MHz を使用した。メモリは 512MB から 768MB である。

OS は、サーバ、クライアント共に UNIX 系の FreeBSD[3] を使用した。バージョンは 6.1#0 であり、特記しない限り基本設定を変更せずに計測した。

サーバとクライアントとの間は、バッファロー製のハブ LSW-GT-8ES[4] を使用し、カテゴリ 6 のイーサネットケーブルで接続した。また、サーバやクライアントは、別回線で制御用のパソコンと接続した。全体の接続を、図 1 に示す。

1.2 ハードディスク

計測に使用したハードディスクは、日立製の Deskstar 7K500 シリーズの HDS725050 KLA360[5] である。ドライブの仕様を表 1 に示す。

計測には `/bin/dd` を用い、バッファを 8MB、転送量を 1GB とした。読み出しは「`dd if=/dev/ad8 of=/dev/null bs=8M count=128 skip=`ディスク位置`*128`」、書き出しには「`dd if=/dev/zero of=/dev/ad8 bs=8M count=128 skip=`ディスク位置`*128`」を実行し、その転送速度を計測した。ディスク内の位置に対する転送速度の変化を図 2 に示す。

表 1: ハードディスクの仕様

Label capacity (GB)	500
Bytes per sector	512
Number of sectors	976,773,168
Total logical data bytes	500,107,862,016
Interface transfer rates (Mb/s)	300
Data buffer size (MB)	16
Rotational speed (rpm)	7200
Mechanical positioning performance (ms)	8.2
Single Track Seek Time (ms)	0.8
Full stroke seek time (ms)	14.7
Latency Time (ms)	4.17

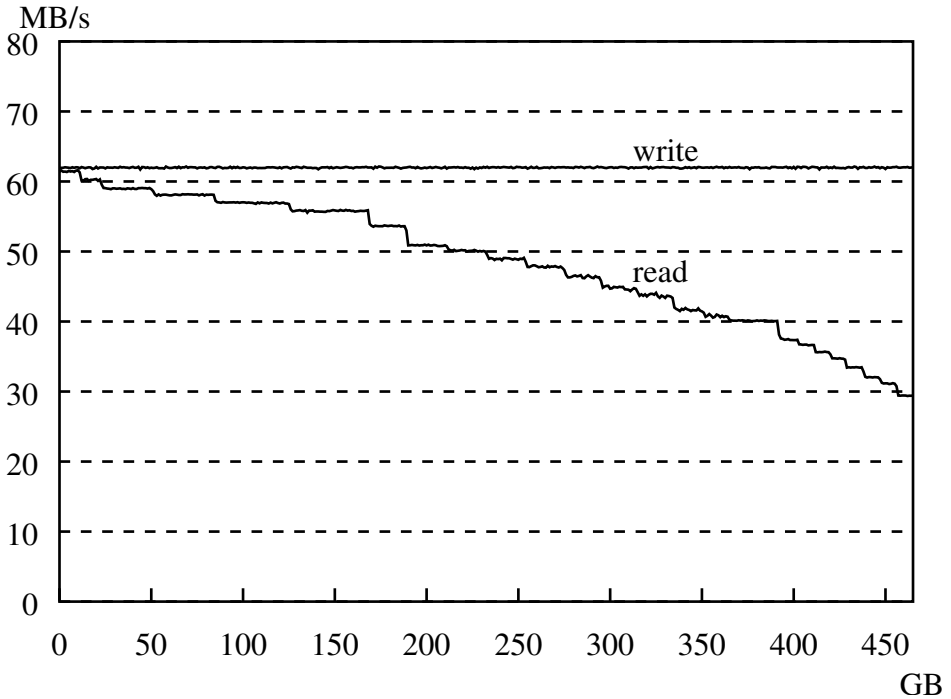

図 2: 位置と転送速度

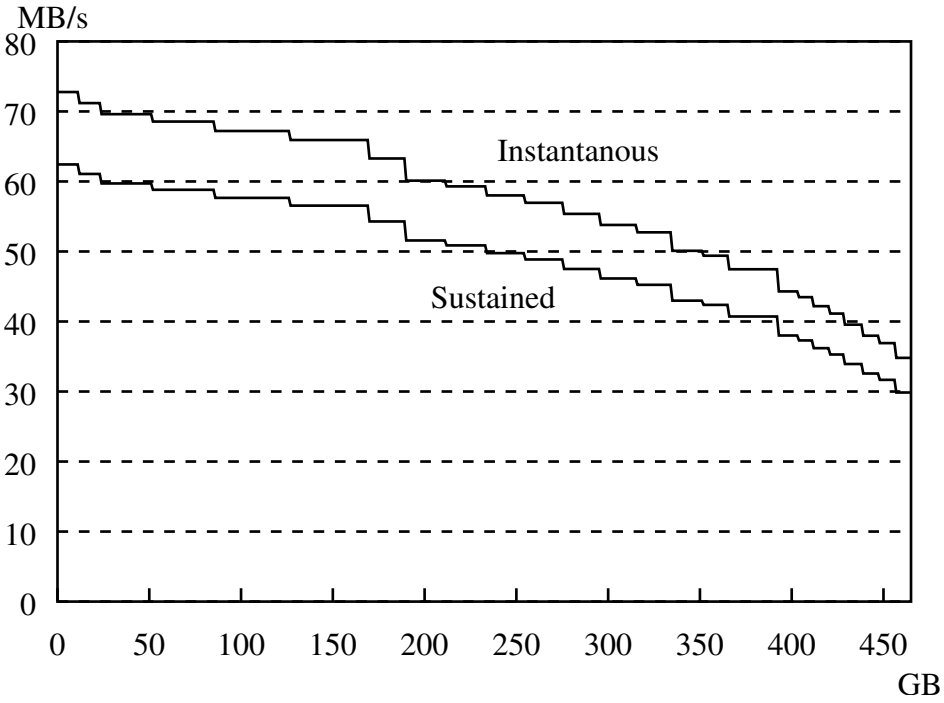

図 3: 読み出し転送速度（計算値）

書き込み速度は、ディスク内の位置にかかわらず、ほぼ一定である。これは、書き込みは OS やディスクドライブがバッファリングしているためであると推察される。しかし、読み出し速度は、ディスク内の位置によって大きく変化している。これは、ディスクの外周と内周での記録密度の違いがあらわれていると推察される。そこで、7K500 シリーズのデータシートより、読み出し速度の理論値を計算した。ディスクは 30 のゾーンにわけられ、各々のゾーンにおいてシリンダあたりのセクタ数が異なっている。シリンダあたりのセクタ数を s、セクタあたりのバイト数を b(Bytes)、回転数を r(rpm) とすると、データ転送の速度 d(Bytes/s) は $d = s * b/r * 60$ となる。計算結果を図 3 に示す。

ヘッドの切り替え時間などを考慮して計算した値 (Sustained) は、実測値とよく一致している。したがって、基本的なソフトウェアである `dd` により、ディスクの読み出し転送速度が計測できることが判明した。

1.3 ネットワーク

ネットワークの性能については、クライアントからサーバに対して`/sbin/ping` コマンドにより ICMP(Internet Control Message Protocol) メッセージを繰り返し送り、その通信速度を、サーバにおいて`/usr/bin/systat` コマンドにて計測した。

ICMP メッセージは８オクテットのヘッダと、任意長のデータからなるが、イーサネットの１フレームが最大 1500 オクテット (Byte) のデータより構成されることから、ICMP メッセージのデータ長を 1472 オクテットとすると、IP ヘッダの 20 オクテットとあわせて、1500 オクテットのイーサネットフレームを構成することとなる。

IP header (20 octets)	ICMP header (8 octets)	Data (1472 octets)

しかし、ICMP メッセージに応答しなければならないことを考えると、単純な通信速度を計測するには、通信量に比して ICMP メッセージへの応答に必要な時間が無視できるように、データ長をなるべく大きくすることが必要である。そこで、データ長を 65112 オクテットとした。これにより、ひとつの ICMP メッセージが 1500 オクテットのイーサフレーム 44 個として通信される。

IP header (20 octets)	ICMP header (8 octets)	Data (1472 octets)
IP header (20 octets)	Data (1480 octets)	
:		
IP header (20 octets)	Data (1480 octets)	

ところで、OS のセキュリティ機能として、大量の ICMP メッセージには反応しないよう設定されている。今回の測定では、この機能により計測が妨害されないように、サーバ、クライアントともに、システム値 **net.inet.icmp.icmplim** を 65535 に設定した。

測定は、クライアントから「`/sbin/ping -fs 65112` サーバ」というプロセスを起動し、サーバ側で「`/usr/bin/systat -if 1`」により計測した。クライアント側のプロセス数による転送データ速度の変化を図 4 に示す。

11 プロセス（64MB/s）をこえると、ネットワークが飽和することがわかる。しかし、これだけではサーバ側が飽和しているのか、クライアントの能力によるものか、途中のネットワーク環境によるものかが判別できない。そこで、別のクライアントからも ping プロセスを起動して、さらに計測した。

さらに 8 プロセス以上を追加すると、ネットワークは 108MB/s で飽和した。また、別クライアントからさらに ping プロセスを起動しても、飽和状態は変化しなかった。イーサネットでは、１パケットにつき 21 オクテットの信号が付加されるので、1500 オクテットのイーサネットフレームを連続して送出したとすると、IP パ

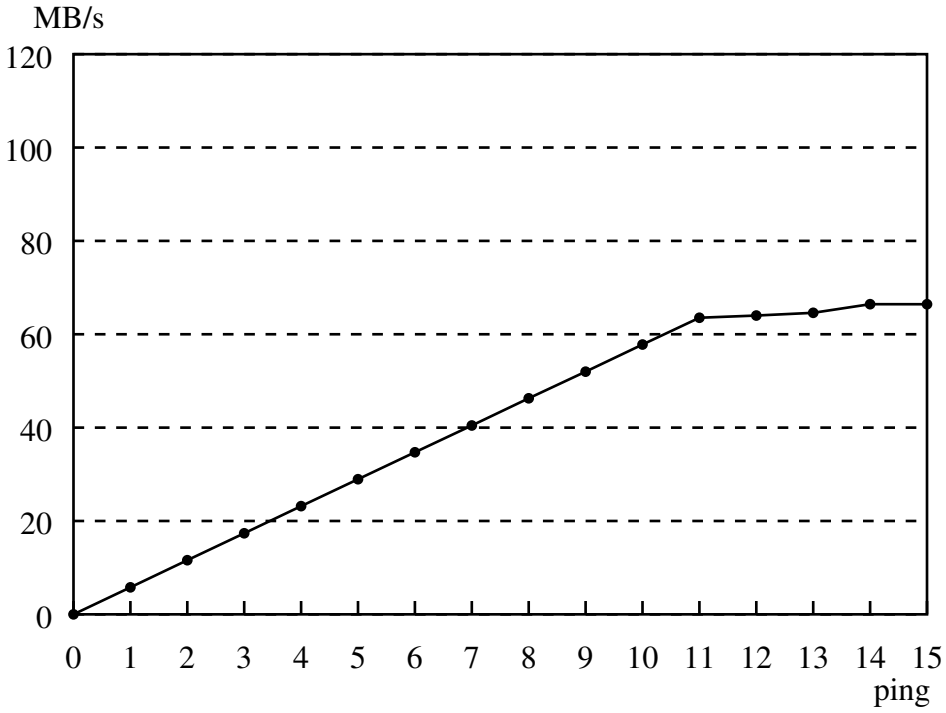

図 4: プロセス数とデータ転送速度

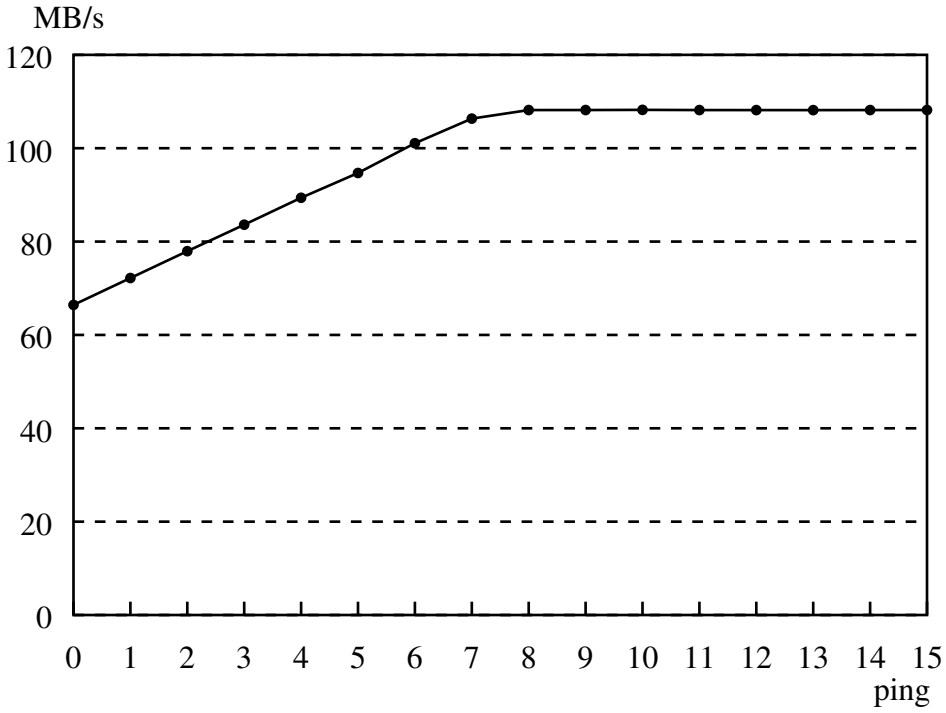

図 5: 追加プロセス数とデータ転送速度

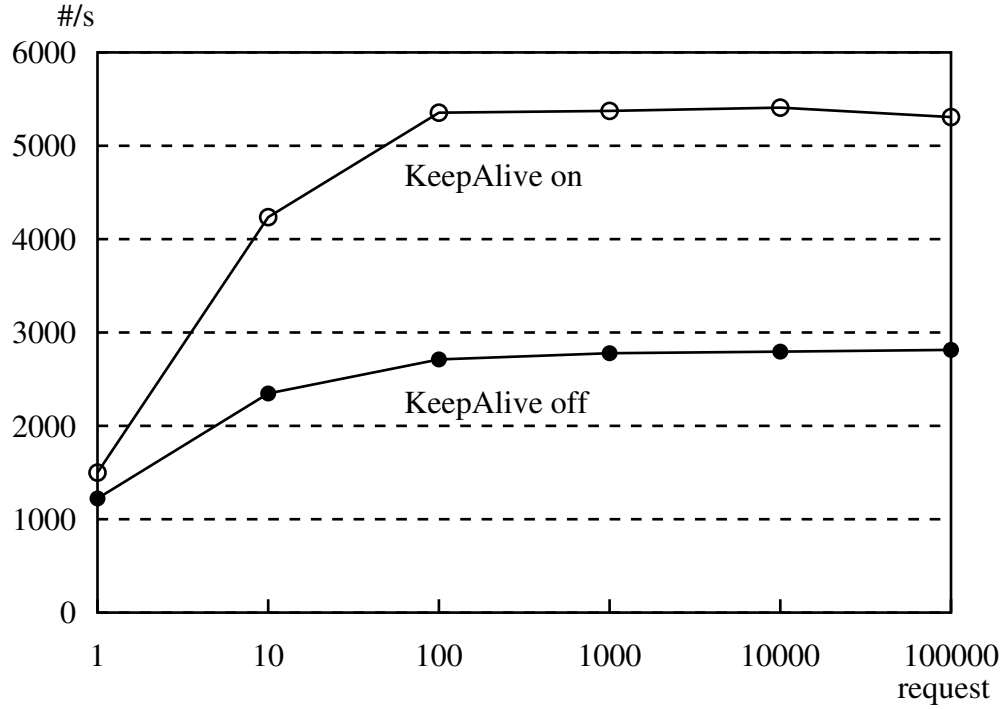

図 6: KeepAlive の有無とリクエスト数によるサーバ速度の変化

ケットのデータは、$10^9/8 * 1480/1521 = 116$MB/s の転送速度となる。実験では、最大でその 93%の値を示しており、ほぼ限界の性能が出ていることがわかった。

さらにハブを交換しても、飽和状態に差はみられなかった。このことから、市販のハブは十分な性能を有しており、ハブがボトルネックとはならないことがわかった。

1.4 サーバ

Web サーバのソフトウェアには、Apache/2.2.2[(6)] を使用した。設定は初期値のままであり、マルチスレッドは使用していない。測定にはサーバ上で ApacheBench (`/usr/local/sbin/ab`) を使用した。

まず、必要な測定数を知るために、KeepAlive の有無それぞれについて、リクエスト数を変化させて「`ab -k -n` リクエスト数 `localhost/`」、「`ab -n` リクエスト数 `localhost/`」を実行した（図 6）。

この結果、リクエスト数は 1000 で十分であるが、KeepAlive の有無による差異が 2 倍近くあることがわかった。

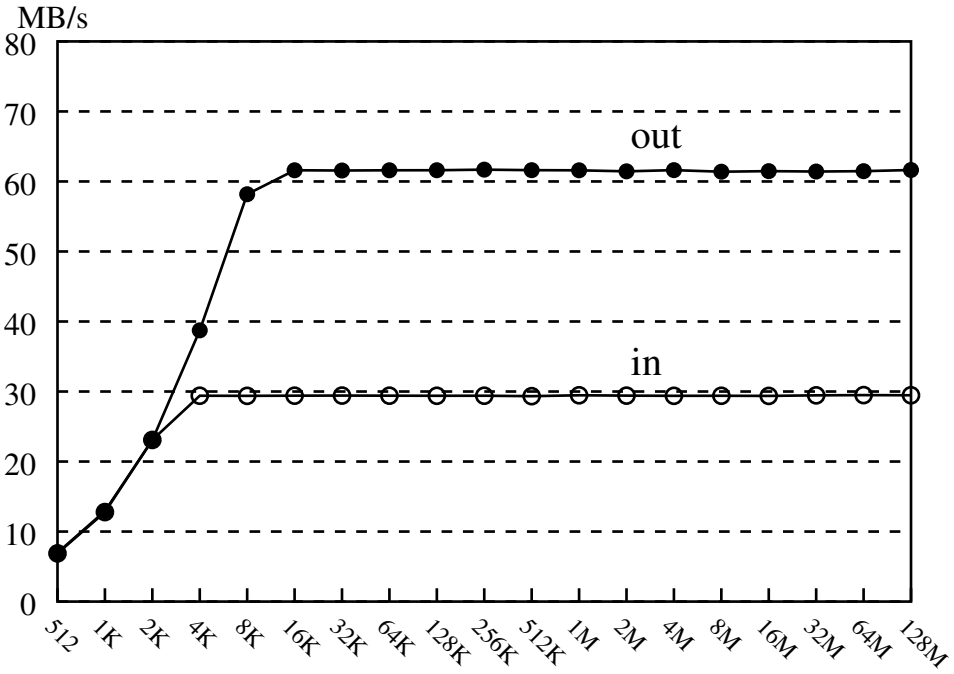

図 7: データ転送量と読み出し速度

2 詳細実験

2.1 データサイズ

2.1.1 ハードディスク

ハードディスクの読み込み速度について、1 回のデータ転送量の違いによるデータ転送速度の変化を `dd` を用いて計測した。計測は、ディスク最外周 (out) と最内周 (in) についておこなった。結果を図 7 に示す。

1 回の読み出しが 8KB 以下では、システム処理の時間が無視できなくなり、データの連続読み出しができなくなる。その結果、読み出しが不連続となり、速度が低下していることがわかる。それ以上のデータサイズでは、データシートからの計算値と合致していることがわかった。

2.1.2 サーバ

データ量の変化に対する応答速度とデータ転送速度を計測した（図 8)。データは 512B から 1GB まで、乱数を使って生成したものをサーバの DocumentRoot に置いたものを使用した。転送速度は ApacheBench(`ab`) と `systat` の両方で計測した

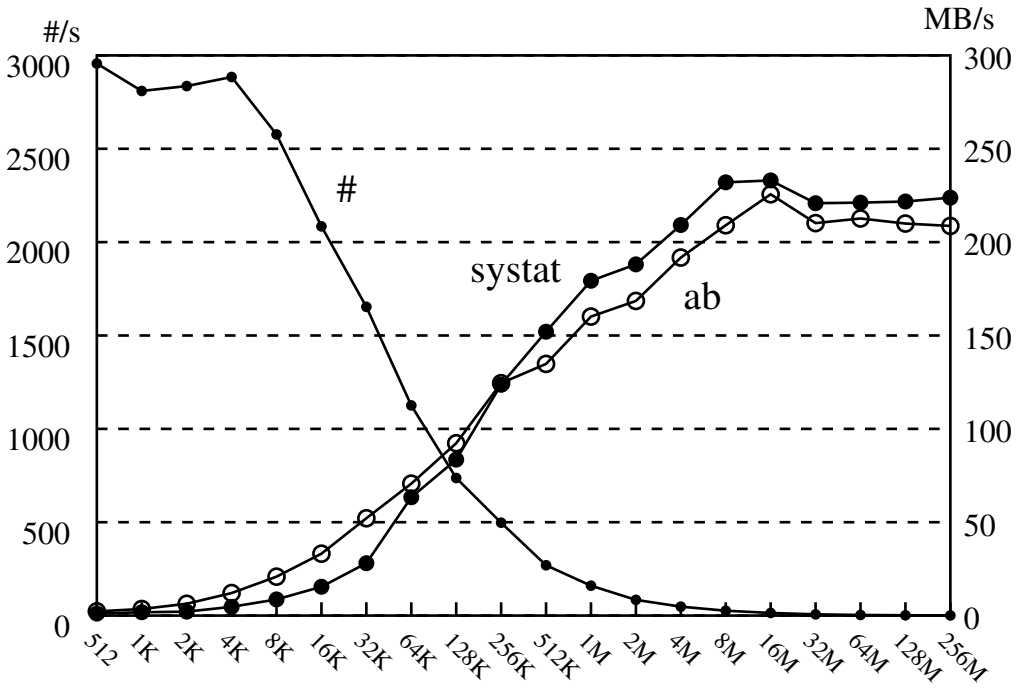

図 8: データ量と応答回数、転送速度

が、大きな差異はみられなかった。ネットワーク経由ではないので、最大 230MB/s の転送速度が得られた。

データ量が小さいときは Web サーバの応答性が支配的であり、データ量が大きくなるとネットワークの飽和がボトルネックとなることがわかった。

KeepAlive 有りの場合は、転送速度の変動がはげしく、ApacheBench による平均転送速度と、`systat` によるピーク値の差が目立つ。

2.2 並列リクエスト

ApacheBench には、リクエストを複数同時におこなう機能 (`-c`) もある。同時アクセス数を 100 にして測定した（図 10)。

KeepAlive の有無によっては転送速度に大きな変化はみられないが、小さいデータサイズのときの応答性は KeepAlive により改善された。

2.3 総合実験

クライアントからサーバに対して ApacheBench を実行した結果を、図 11 に示す。KeepAlive は off で、並列リクエスト (`-c`) の有無それぞれについて計測した。

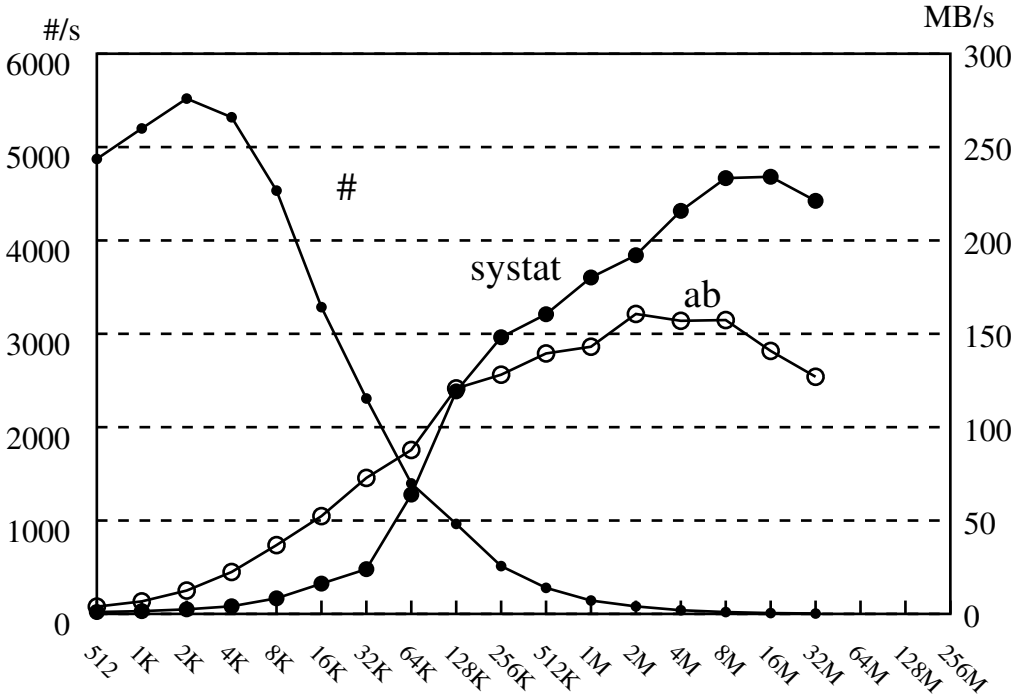

図 9: KeepAlive ありの場合

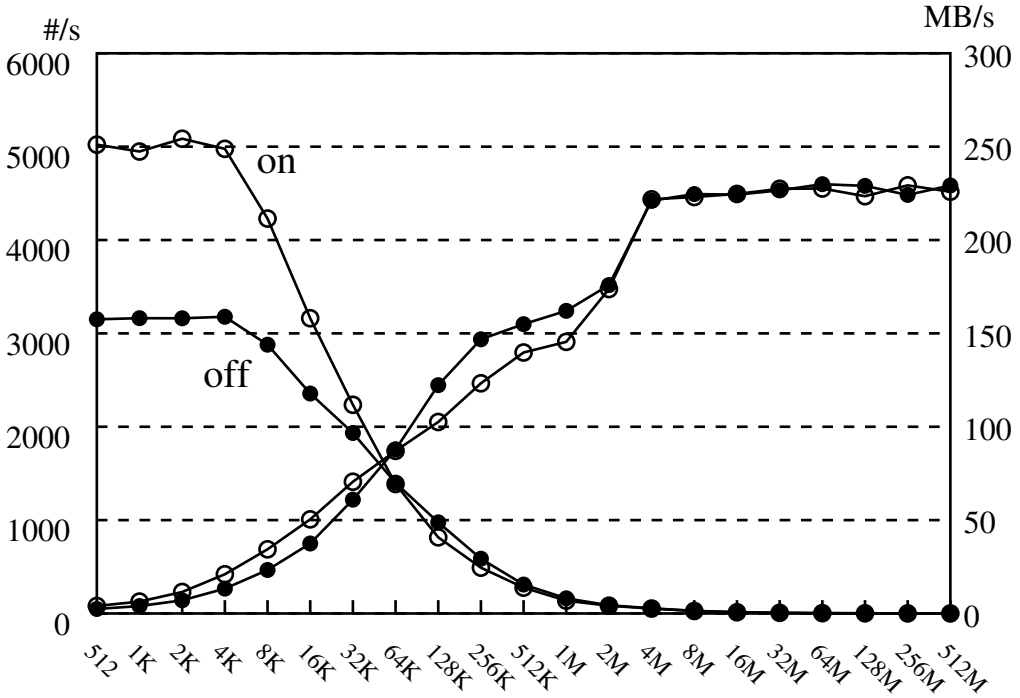

図 10: 並列リクエスト数 100 の場合

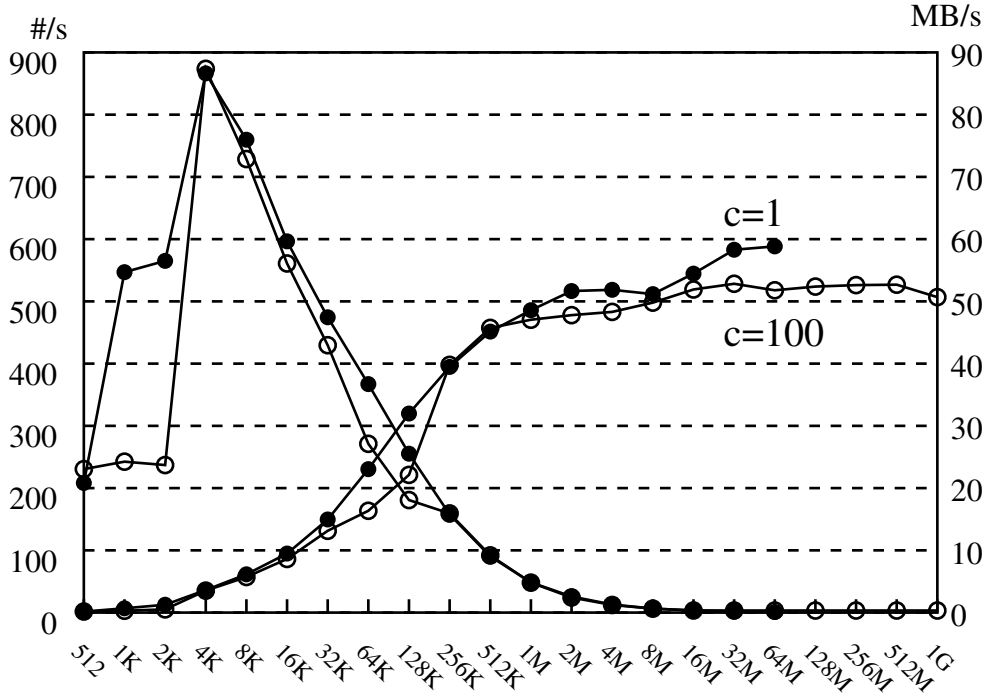

図 11: ネットワーク経由の場合

データサイズが大きくなると、60MB/s で転送速度が飽和している。これは 1.3 節で計測したクライアントの能力 64MB/s という値に近い。64MB/s まで伸びないのは、1.3 節とは違って、イーサネットの 1 フレームが 1500 オクテットだけではなく、様々な大きさのフレームが生成されることから、全体としての転送速度の低下がおこっていると推測される。

また、ハブを 100BASE-T のものに変更して計測したものを図 12 示す。この環境では 11MB/s でネットワークが飽和していることがわかった。

データ数が小さいときは、サーバプロセスの生成時間やネットワークの遅延時間が原因で、計測値の変動が大きくなっていると推察される。

3　まとめと課題

3.1　測定ソフト

ハードディスクの基本性能を計測するには、データサイズに注意すれば、dd を使用することに問題はない。

ネットワークの計測については、ひとつのクライアントから ICMP メッセージを送るだけでは不十分である。しかし、複数クライアントから複数の ping プロセ

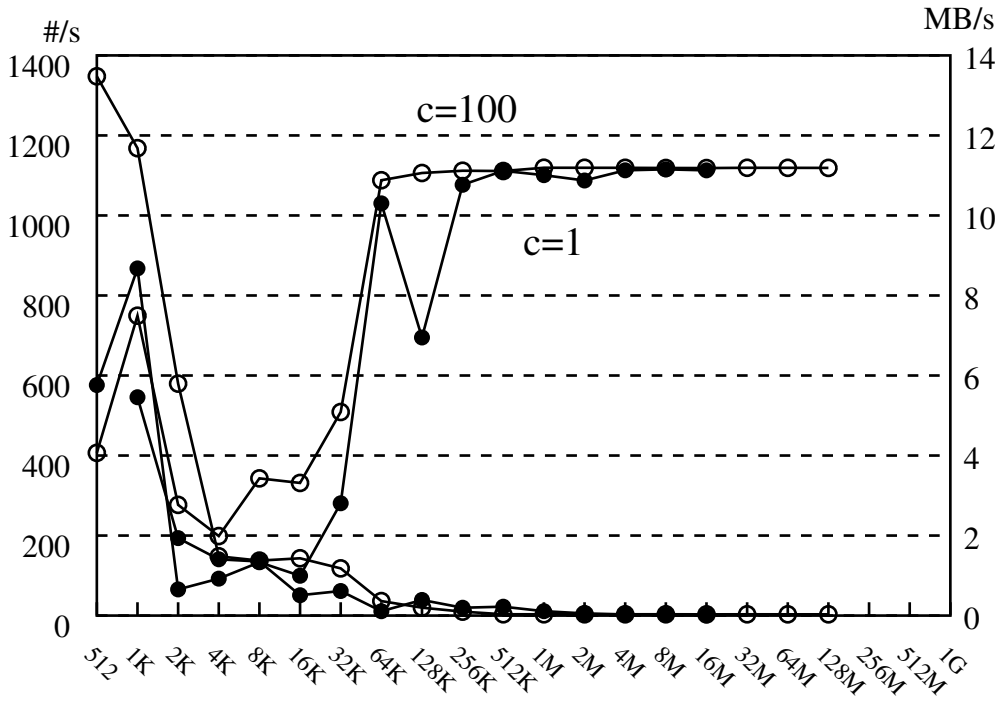

図 12: 100Mbps ハブの場合

スを実行すれば十分であり、それを `systat` で計測するだけで、特別なソフトウェアを準備する必要はない。

ApacheBench は様々な設定での計測ができるが、測定数値がオーバーフローしたときに表示が狂うバグがある。また、転送速度がピーク値でなく平均値を表示するので、環境の変動により誤差が生じる。データ量が大きい場合に異常終了して、計測値が得られない場合もあった。さらに、複数のクライアントから同期してアクセスすることができないため、大規模な実験には不適当である。これらのことを考慮して、ApacheBench を改良するか、別ソフトウェアを開発する必要がある。

3.2 クライアントの能力

クライアントには、サーバと同等もしくはそれ以上の処理能力が必要である。特に、大量のリクエストを同時に送出するためには、ネットワークまわりのハードウェアとソフトウェアに、十分な能力が要求される。

3.3　ファイルシステムの影響

今回の実験では、サーバ側でファイルシステムから読み出されるデータは一定のものであり、OS の内部でバッファリングされることが期待される。そのため、ファイルシステムの影響は測定にあらわれない。

ただし、ハードディスクのパフォーマンスから推察するに、ファイルシステムを構成するときの読み出し単位を 16KB 以上にすることで、ファイルシステムの効率が向上することが期待される。

3.4　パフォーマンスチューニング

データサイズが大きい場合、ネットワークがボトルネックとなる。この場合には、ソフトウェアのチューニングではなく、ハードウェアの換装が主眼となる。

逆に、データサイズが小さい場合は、ソフトウェアの設定をチューニングすることにより、サーバのパフォーマンスを改善することができる。

注

(1) ASUS, P5LD-2V,
http://jp.asus.com/products4.aspx?modelmenu=1&model=575&l1=3&l2=11&l3=0

(2) インテル, PRO/1000 GT,
http://www.intel.com/network/connectivity/products/pro1000gt_desktop_adapter.htm

(3) FreeBSD,
http://www.freebsd.org/

(4) バッファロー, LSW-GT-8ES,
http://buffalo.jp/products/catalog/item/l/lsw-gt-8es/index.html

(5) 日立グローバルストレージテクノロジーズ, Deskstar 7K500,
http://www.hitachigst.com/portal/site/jp/menuitem.9c6856e3c11793518797c532aac4f0a0/

(6) Apache,
http://httpd.apache.org/