Vol.2010-DPS-142 No.7
Vol.2010-CSEC-48 No.7
2010/3/4

汎用機器を用いたネットワークトラフィック解析を実現するパケットキャプチャリングシステムの最適化手法

塚原康仁[†]　遠峰隆史[†]　杉浦一徳[†]

本研究ではネットワークトラフィック解析を行うためのパケットキャプチャリングシステムを汎用 PC で実現するための最適化手法について議論する．
具体的に，tcpdump ツールを使用した場合の条件下で，raid0 やカーネルのチューニングなどを施した場合に，パケットのドラップ量の変化をもとに最適なチューニングを検討していく．

The Tuning Method of Packet Capturing System to Put Analyzing the Network Traffic Data with General Computer into Reality

Yasuhito Tsukahara[†] Takashi Tomine[†] kazunori Sugiura[†]

In this paper, we propose the tuning method of packet capturing system to put analyzing the network traffic data with general computer into reality. Concretely, we observe how the amount of dropped packet data shifts by utilizing RAID0 and Kernel tuning if we use tcpdump tool..

1. はじめに 【*の文字書式「隠し文字」】

本論文ではパケットキャプチャのパフォーマンス向上を目指した，チューニング手法の提案を行う．tcpdump ツール[1]といった広く普及したツールを使用し，パケットキャプチャを行う場合，パケットロスやパケットドロップが生じる．これは，ネットワークの監視やネットワークトラヒック解析などを行うにあたって，信頼性の高いデータを取得するのに問題となる．本論文では一般的に利用されているパケットモニタリングツールを利用してのパケットキャプチャを行う場合に，パケットロスやパケットドロップによって必要とするデータを取得できなといった問題を解決するためのチューニング手法の提案を行う．

また，一般的な PC を用いてパケットキャプチャを行った場合のパケットロス，パケットドロップの改善のチューニング手法でもある．tcpdump ツールなどを利用して，ノート PC などでもパケットキャプチャは容易にできる．しかし，通信速度の向上などによってネットワークトラヒック量は増加している．したがって，通常の PC では性能上，帯域幅の大きいネットワークでパケットキャプチャを行う場合，パケットロスやパケットドロップすることなくパケットキャプチャすることは難しいのが現状である．このような問題を解決し，通常の PC でも提案するチューニング手法を行うことで，パケットロス，パケットドロップすることのないパケットキャプチャ専用の汎用機器を作り上げることが本論文の目的である．

一般的なパケットモニタリングツールを利用して，一般的な PC でパケットキャプチャを行う場合にパケットロス,パケットドロップを引き起こす原因を特定していく．そして，その原因を解決する手法を議論していき，その解決策をチューニング手法とする．

2. 現状分析

一般的な PC を用いたパケットキャプチャリングシステムのチューニング手法の提案を行っていくにあたって，基準となる PC の性能を提示する．次に，tcpdump ツールを利用したパケットキャプチャでそのパフォーマンスを測定する実験方法を明示し，実際にチューニングを行っていない状態でのパフォーマンスの計測をおこなう．最後に，計測結果を元にパケットキャプチャ時の問題点を検討していく．

2.1 標準状態での PC の性能

パケットキャプチャのパフォーマンスを向上させるためのチューニングを施してい

[†] 慶應義塾大学大学院メディアデザイン研究科　神奈川県横浜市港北区日吉４－１－１

Vol.2010-DPS-142 No.7
Vol.2010-CSEC-48 No.7
2010/3/4

った場合, その効果を測定するために基準を明確化する. この状態を標準状態とする. CPU は Intel 社製品の core i7 を使用する. ハードディスクは SATA を利用し, 容量 750GB, 回転数 7200rpm, キャッシュメモリ 16MB の UMAX 社製品のものを利用する. ハードディスクのインターフェスは Serial ATA300 である. 使用するオペレーティングシステムは Debian GNU/Linux 5.0.3 AMD64[2]. カーネルは 2.6.32.4 を利用. ファイルシステムは ext3（third extended file system）, その他本論文においてのチューニングに, 関係するものを含め表 1 に示す.

OS	マザーボード	HDD	HDD インタフェース	CPU	メモリ
Debian 5.0.3 AMD64	ASUS P6T	Seagate Barracuda7200.12 SATA	Serial ATA 300	Intel corei7	UMAX cetur TCDDR3-3GB-1600OC

表 1 汎用機器で利用しているもの詳細

また, 利用している NIC で最低限必要とする HDD のスループットを考える. 一般的に, 10G, 1G, 100M である. これらの帯域毎で必要とする HDD の値は表 2 に示す.

NIC 帯域	100Mbps	1Gbps	10Gbps
HDD	12.5MB/s	125MB/s	1.25GB/s

表 2 NIC の対応する帯域毎で HDD が必要とするスループット

2.2 パケットキャプチャのパフォーマンス計測の実験方法

パケットキャプチャにあたってのパフォーマンスの測定実験の方法に関して述べる. 必要な機材は PC2 台, Ethernet である. PC1 台をパケットキャプチャ用とし, 1 台をパケットジェネレーターとする. 接続方法は, それぞれの PC にプライベート IP アドレスを割り振り, これら 2 台の PC を Ethernet で接続する. そして, C 言語にて作成したパケット生成プログラムを用意し, パケットジェネレーターとして利用する. このプログラムは UDP で指定した IP アドレスへ入力した帯域分のデータを送信するものである. 一般的なジェネレーターツールなどもあるが, 今回はこのプログラムにて実験を行う. 帯域幅は ifstat ツールを使って計測した結果, 最大で約 100MB/s である. この最大値はプログラム上 μ sec までしか扱えないようになっているからである. 帯域幅はコマンド上で入力可能であり, 制御することができる. また, パケットキャプチャ用の機器は tcpdump ツールにてパケットをキャプチャする. ネットワークインターフェースを eth0 とすると,

#tcpdump –i eth0 –n –s 0 –w hoge

でキャプチャする. ドロップ量は, tcpdump ツールが出力する値をドロップ量とする. また, パケットのロスト量は, netstat にて,

#netstat –i

で RX-DRP の数値を参照する. キャプチャ時間は 1 分とし, 送信量はキャプチャ時間分の合計値とする. 実際その間のパケットのドロップ量を計測する. 以上が, パケットドロップ量, パケットロス量の計測の実験方法である.

2.3 デフォルト状態でのパケットドロップ, パケットロストの推移と問題点

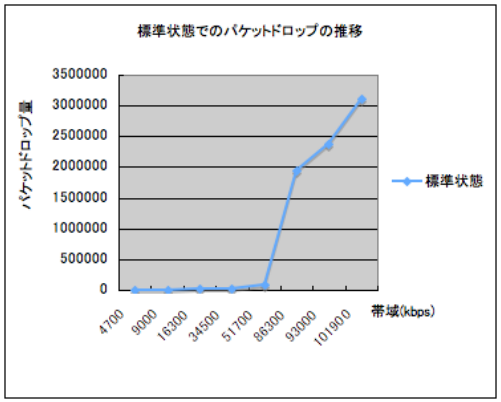

図 1 デフォルト時のパケットドロップ量

図 1 のグラフはデフォルトでのパケットドロップ量の推移である. 縦軸はパケットのドロップ量, 横軸は kB/s である. 50MB/s を超えたあたりからのロスト量が目立つ. そして, 50MB/s から 100MB/s にかけてのドロップ量の増加が著しい. よって, 50MB/s からのパケットドロップ量を減らしていくことが課題である. また, パケットロス量に関して netstat の RX-DRP からはパケットロスは見られなかった. よって, 50MB/s 以上の帯域でのパケットドロップ量を減らすことを目標とする.

Vol.2010-DPS-142 No.7
Vol.2010-CSEC-48 No.7
2010/3/4

次に，この課題を解決していくにあたって，ボトルネックを特定していく．一般的なボトルネックとして以下の項目があげられる．

① CPU 使用率
② メモリ使用量
③ ディスク I/O

①から順にベンチマークにて評価を行う．

2.4 CPU 使用率

まず CPU 使用率の評価を行う．CPU の使用率が 100%に近い場合は CPU の処理能力の問題となる．逆に CPU の使用率が低ければボトルネックではなく，原因別の箇所だと判断することができる．

idle
66.9%

表 3 top ツールでの CPU 使用率

表 3 に CPU の idle 部分を示す．ジェネレーターで約 101MB/s をキャプチャ用機器に送信し，tcpdump を用いて取得しているときの CPU 使用率である．top にて取得した値である．idle が 66.9%なので，単純計算で，33.1%が使用率となる．

2.5 メモリ使用量

次にメモリの使用状況を見てみる．もし，物理メモリの容量を常に超えるような場合はメモリの容量が足りないことが原因であると言える．デフォルト状態で 3GB のメモリ容量なので，それ以上の場合かそれに近い値の場合がメモリ容量がボトルネックであることを指す．

Total	Used	Free	Shared	Buffers	Cached
3085284	548316	2536968	0	12176	463300

表 4 free ツールによるメモリ使用状況

表 4 は free ツールで取得したものである．実際に使われているメモリサイズは 535.46MByte である．

2.6 ディスク I/O

最後にディスク I/O に関して考察していく．ディスク I/O が問題の場合，ジェネレーター機器から送信される帯域幅を処理できるスループットでないなら,ディスク I/O がボトルネックであると言える．実際に，ジェネレータが送信できるのは約 100MB/s なので，HDD のスループットは 100MB/s を少なくとも超える必要がある．

ここでもう一度，HDD の性能を確認する．SATA の 750GB，7200rpm でキャッシュ 16MB である．この HDD 一台における，書き込み速度のベンチマークをとりパフォーマンス改善が必要かを検討していく．

実際にベンチマークをとる前に HDD の性能計測実験に関しての方法をまとめる．ベンチマークツールは多数利用できるが，今回は time コマンドと dd コマンドを使用する．利用理由は，実際にディスクに書き込む時間を計測できるので，パケットを実際にディスクへ書き込む状況に近いからである．

time はコマンドによって使用されたリソースの情報を表示するコマンドである．したがって，time を使ってコマンド実行時間およびディスクへの書き込み時間を計測するために利用する．他方，dd コマンドはファイル操作で扱うコマンドである．dd コマンドを使ってファイルを生成し，その生成されたファイルをディスクへ書き込むのに利用する．

そして，実際に書き込んでそのディスク I/O のパフォーマンスを測るのは，1 MB を 1000 回書き込む（計 1GB）時，1GB を 1 回書き込む時，2GB を書き込む時の 3 通りで HDD のパフォーマンスをとる．以下順にコマンドを記載する．

1 MB1000 回

```
#time dd if=/dev/zero of=/home/hoge ibs=1M obs=1M count=1024
```

1GB1 回

```
#time dd if=/dev/zero of=/home/hoge bs=1000M count=1
```

2G1 回

```
#time dd if=/dev/zero of=/home/hoge bs=2000M count=1
```

上記の方法で取得した，ベンチマークは以下表 5 である．

Vol.2010-DPS-142 No.7
Vol.2010-CSEC-48 No.7
2010/3/4

サイズ	回数	実際の書き込み時間（秒）	スループット（MB/s）
1M	1024	19.42	51.485
1G	1	16.69	59.916
2G	1	34.96	57.21

表 5 HDD 性能計測結果

１０回それぞれの手順を行った．そこから平均値を算出した結果を表 5 に示す．50MB/s~60MB/s 未満である．パケットドロップ量が増加したのは，ジェネレーターが，帯域幅５０Mbps 以上を送信したときに発生したことをふまえると，HDD のスループットは１つの問題点であると言える．よって，まずはディスク I/O のチューニングが必要である．

2.7 現状分析のまとめ

本章では，汎用機器におけるパケットキャプチャの考察を行ってきた．計測実験の結果からパケットロス，あるいはパケットドロップの可能性が見受けられた．そして，netstat –i の結果からパケットロスの可能性は低いことが明らかになった．そして，一般的なボトルネックの３つ部分からディスク I/O が問題点として指摘できる．次章からディスク I/O のチューニングを行い，実際にパケットドロップの改善が見られるかを中心に考え，ジェネレーターからの１００MB/s までをパケットドロップすることなくパケットキャプチャを実現するチューニング手法を考えていく．

3. パフォーマンス向上のためのチューニング

2 章を踏まえ，本章では実際にチューニングを行っていく．一つの問題点としてあげられるディスク I/O の改善から着手する．そして，パケットキャプチャのパフォーマンスを計測しながら評価をし，最適なパフォーマンス手法を提案していく．

3.1 ディスク I/O のチューニング

ディスク I/O のチューニングを行っていく．実際に利用する手法は RAID 0（ストライピング）である．RAID（Redundant Arrays of Inexpensive Disks）は複数の HDD をまとめて１台のハードディスクとして管理する技術である．そして，RAID 0 は RAID のレベルの１つである．複数のディスクに均等にデータを振り分け，同時並列でデータを書き込んでいくシステムである．データを振り分け，同時並列でデータを書き込んでいくので，ディスクの数が多くなれば（配列数），ディスク I/O のパフォーマンスは向上する．

本論文では，汎用 PC での RAID 0 を構築する．したがって，Linux kernel で実現可能なソフトウェア RAID を利用する．7200rpm，750GB の HDD で RAID 0 を行う．その他，前提としてあげた性能は同じものである．

続いて，RAID 0 を組んだ手順を示す．

```
#fdisk sdb（以下 fdisk 内で）
Command(m for help): n
primary partition: p, Parition number:1
First cylinder: 1,
Last cylinder or +size or +sizeM: +M75000
Command(m for help): t
Hex code(type L to list codes): fd

#mdadm –C /dev/md0 –l 0 –n 2 –f /dev/sd[bc]1
#mkfs –t xfs /dev/md0
#mount /dev/md0 /dump
```

このような手順にて行った．そして，RAID 0 の配列数の増加とともにディスク I/O のパフォーマンス向上を評価するために，配列数は 2，3，4 と増やしていき，パケットドロップ量の減少具合を評価していく．

ここで，適用したファイルシステムに関して述べておく．Debian をインストールした際に Debian 関係のファイルシステムは ext3 を利用しているが，RAID 0 を適用したディスクには xfs(The eXtended File System)[3]を利用した．ext3 と xfs の両者はジャーナリングに対応したものである．大きな違いは，xfs の方が，ext3 よりも大容量のファイルとパーティションサイズに対応している点である．xfs のこの特徴は，パケットキャプチャリングシステムに適している．パケットキャプチャは大容量のデータと大容量のハードディスクを必要とする．よって，ext3 では対応できない容量のものでも，xfs では扱うことができる．実際に，ext3 と xfs を比較すると，ファイルシステムの最大サイズは ext3 が 16TB に対して，xfs は 16EB である．また，ファイルの最大サイズは ext3 が 4TB に対して，xfs は 16TB である．数字からも明らかで，xfs の方が大容量に適している．よって，xfs を適用した．

ここまでは，ディスク I/O のチューニングとして RAID 0 を適用したことを述べてきた．次項は RAID 0 適用後の評価を行っていく．

Vol.2010-DPS-142 No.7
Vol.2010-CSEC-48 No.7
2010/3/4

3.2 ディスク I/O チューニング後の評価

実際に，パフォーマンスの向上を評価していく．評価実験は，2 章で述べた方法を実際に行う．まず，配列数毎のベンチマークをとり，その後，パケットドロップ評価の計測実験を行う．

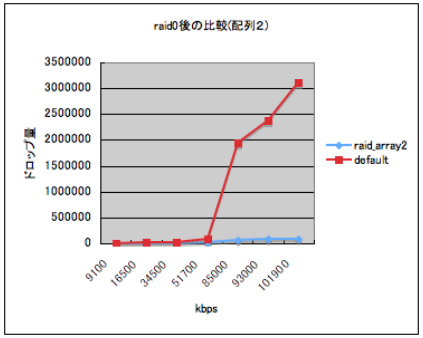

図 2 RAID 0 配列 2 と default の比較

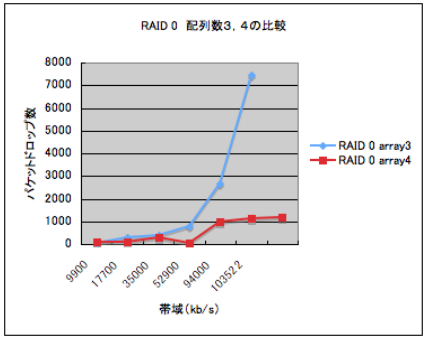

図 3 RAID 0 配列 3 と配列 4 の比較

図 2 は配列 2 とデフォルトの比較である．続いて，配列 3，配列 4 でのドロップ量の推移を見てみる．

図 3 はそれぞれ，配列 3，配列 4 のグラフである．配列数を増やすことによって，パフォーマンスの向上が見られる．デフォルトのドロップ量に対して帯域が１００MB/s の時，減少率はそれぞれ 99.8%，99.9%となった．しかし，帯域が 9 MB/s からパケットドロップが見られた．実際の値をまとめた表を示す，

配列 3							
kB/s	9900	17700	35000	52900	94000	103522	
ドロップ量	33	276	371	767	2684	7438	
配列 4							
kB/s	9900	17700	35000	52900	94000	103522	
ドロップ量	99	88	279	43	1114	1178	

表 6　実際の値

今後の課題はこの微量なドロップ量をなくすことである．

ベンチマークにて評価する．方法は同様で，time コマンドと dd コマンドを使ったものである．ベンチマークは以下の表 7，8，9 である．

データサイズ	回数	実際の書き込み時間（秒）	スループット（MB/s）
1M 1000 回	1024	4.13	267
1G 1 回	1	4.12	248.79
2G　1 回	1	13.06	156.81

表 7 配列 2 のディスク I/O のベンチマーク

データサイズ	回数	実際の書き込み時間（秒）	スループット（MB/s）
1M 1000 回	1024	2.67	383.52
1G 1 回	1	2.61	392.33
2G　1 回	1	9.21	222.37

表 8　配列 3 のディスク I/O のベンチマーク

データサイズ	回数	実際の書き込み時間（秒）	スループット（MB/s）
1M 1000 回	1024	2.22	922.52
1G 1 回	1	2.04	1003.92
2G　1 回	1	6.277	326.27

表 9　配列 4 のディスク I/O のベンチマーク

表より明らかでディスク I/O の性能は向上している．配列 4 でのスループットから，ジェネレーターから 100MB/s の帯域を受けた場合，十分にディスクへ書き込んでいける数値を示しているよって，微量なパケットドロップ量はディスク I/O が原因ではないと推測できる．

Vol.2010-DPS-142 No.7
Vol.2010-CSEC-48 No.7
2010/3/4

3.3 微量なパケットドロップの原因

ディスク I/O 以外で微量なパケットドロップ量の原因を特定してく．パケットドロップ量は微量である．2 章で述べたように，CPU やメモリ，NIC はボトルネックではなかった．そこで，パケットを受信してハードディスクへ書き込まれるまでのプロセスを追っていき，どのフェーズで微量のロストがなされるのかというところ考えていきたい．パケットが入ってくるとまず NIC で受信する．NIC では，パケットロス見られていない．つまり，OS で処理されていることになる．次に，buffer memory にパケットが蓄積されていく．一定以上の容量に達したら，HDD へ書き込まれていく．そこで，この過程で仮にドロップしているならば，考えられることは buffer memory から HDD へ書き込まれるときに割り込み処理が発生し，パケットドロップしている可能性がある[4]．そこで，この可能性を検証してみる．

まず，割り込み処理の頻度に関して考えてみたい．割り込み処理の頻度は timer frequency であるが，この値は Linux kernel では，100Hz，250Hz，300Hz，1000Hz．Hz の値が小さいほど割り込み処理の頻度は少なくなる．ならば，timer frequency の値が小さいほど buffer memory の容量は必要である．buffer memory が一定以上の容量に達すると，HDD へ書き込まれるか，あるいは破棄される．だから，仮に 100Hz にするならば，割り込み処理の頻度が減るので値を大きくしなければ，buffer memory の容量を超えて受信したパケットを破棄する可能性があり，逆に 1000Hz にするならば，割り込み処理の頻度が増加するので，パケットの容量は少なくてよい．そこで，buffer memory から HDD へ書き込まれる段階で微量のパケットドロップが発生しているのを検証するために，実際に timer frequency と buffer memory の関係で，計測実験を行う．この実験を通し，検証を試みる．

3.4 カーネル再構築

検証を行っていくにあたって，timer frequency の値を変更する必要がある．そのため，Linux kernel のリビルドを行う．本論文では，2 章で述べた通り 2.6.32.4 を使用する．手順は以下に示す．

#w3m www.kernel.org
#bzip2 –cd linux-2.6.32.4.tar.bz2 / tar –xvf
#cd /linux-2.6.32.4
#make menuconfig
(timer frequency 変更)
#make
#make install
#make modulesinstalll
#cd /boot
#mkraminitfs –o initrd.img-2.6.32.4 2.6.32.4
#cd /boot/grub
(source.list を改変)
#reboot

そして，これより buffer memory と timer frequency の関係を検証していく．

3.5 timer frequency と buffer memory の関係の検証

Linux-2.6.32.4 の標準状態の timer frequency は 250Hz である．今までの計測実験は２５０Hz で行ってきた．これより，timer frequency の値を変更し実際にパケットキャプチャのパフォーマンスを観察し，適した値を考えていく．

まずは，1000Hz と 100Hz のパフォーマンスを計測する．パケットキャプチャによりどちらが適しているのかを決定したいからである．buffer memory のサイズはデフォルトで net.core.rmem_max=128kbyte で net.core.rmem_default=122kByte である．この値は sysctl にて取得した．そして実際に計測した結果は以下のようである．配列数は３で行った．

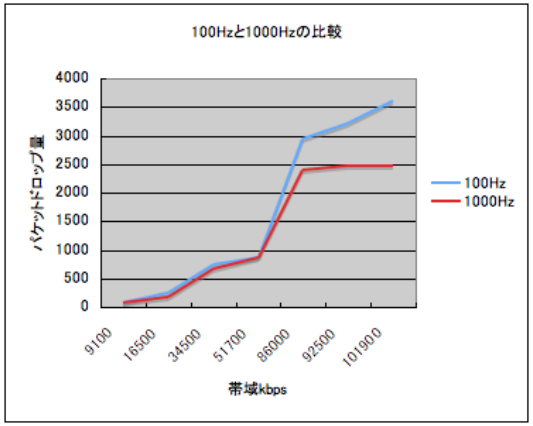

図 4　100Hz と 1000Hz

図 4 をふまえて考察していく．パケットドロップ量は両者ともに微量である．また，250Hz での RAID 0 の配列３でのパケットドロップ量より改善されているまた，Net.core.rmem_max=128kbyte であることから buffer memory サイズは小さいと言える．

Vol.2010-DPS-142 No.7
Vol.2010-CSEC-48 No.7
2010/3/4

だから，buffer memory サイズを増やした時の傾向を見ていくことが適切のように考えられる．したがって，timer frequency を 100Hz として buffer memory のサイズを 2 倍，４倍，８倍と増やしていったときのパケットドロップ量の減少を計測する．

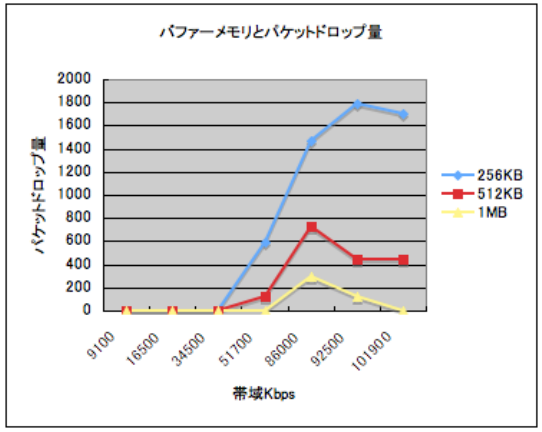

図５　バッファーメモリとパケットドロップ量

図 5 は実際の buffer memory とパケットドロップ量の推移である．データのばらつきはあるものの buffer memory を 2 倍，４倍，８倍と増やしていく毎に微量のパケットドロップの改善が見られた．このパフォーマンスは RAID 0 の配列４よりもよい数値である．しかし，微量なパケットドロップがやはり見られることと，帯域の増加分だけの増加傾向ではなくなった．

3.6 今後の課題

前項では，buffer memory とパケットドロップ量の関係でチューニングを試みた．結果的に RAID 0 で配列数を３にした場合，timer frequency を 100Hz にし，buffer memory のサイズを１M にすることにより，配列数４よりも高いパフォーマンスを示した．しかし，パケットドロップ量とジェネレーターから送信される量の相関関係は見られなくなった．

この原因に関して考察してみる．xfs はジャーナルログ書き込み時に，7 段階のトランザクションを有している[5]．トランザクションのアロケーションから始まり，ログスペースの確保，リソースのロック，リソースの修正およびオペレーションの明示がなされ，コミットトランザクション，そしてジャーナルログの完了である．ジャーナルログの書き込みが完了した段階で，実際にディスクへの書き込みがなされる．このトランザクション間でパケットドロップが起きている．つまり，ディスクへの sync の発生時が原因である．

よって，今後の課題としてこの sync が起きたときにどのようにしてパケットドロップを防ぐかということがあげられる．この問題を解決することによって 100MB/s のパケットキャプチャはパケットロスト，ドロップすることなく，容易に行うことができるようになる．

4. おわりに

tcpdump ツールを利用した場合に，パケットロスおよびパケットドロップをする問題があった．本論文では，その問題を解決するために汎用機器で行うことができるチューニング手法を提案するべく，CPU，メモリ，HDD のベンチマークなどを見ながらチューニングを行い，パフォーマンスの向上を評価してきた．現時点で，RAID 0 で配列数を３とし，buffer memory を 1M，timer frequency を 100Hz とすることで 100MB/s までの帯域で 99%以上パケットドロップおよび，パケットロスを防ぐことが可能である．しかし，sync の割り込みが発生した際に，微量のパケットドロップがおこる．この問題を解決することが課題である．

5. 参考文献

1) tcpdump http://www.tcpdump.org/
2) Debian JP Project http://www.debian.or.jp/project/organization.html
3) Eduardo Diliendo, Takechika Kunimasa, IBM 「Linux Perfomance and Tuning Guidelines」
4) Luca Deri, NETikos S.p.A. : http://luca.ntop.org/ 「Improving Passive Packet Capture: Beyond Device Polling」
5) Adam Sweeney, Silicon Graphics Proprietary http://oss.sgi.com/projects/xfs/ 「xFS Transaction Mechanism」

† 慶應義塾大学メディアデザイン　神奈川県横浜市港北区日吉４−１−１