DNS Demystified
あなたの知らないDNSの世界

2004/07/23 JANOG14 @ 宮崎
株式会社インターネットイニシアティブ
山本 功司 koji@iij.ad.jp
春宮 健二 haru@iij.ad.jp

あなたの知らないDNSの世界…

- いや、そんな大げさな話じゃないんですけどね
  - 季節柄、涼しげなタイトルがいいかな、と^^;
- Authoritative サーバーを正しく設定・運用しましょうという話はよく聞くけど
  - JPNIC、JPRS、DNSQCTF の皆さんの啓蒙活動等
- キャッシュサーバー(caching server)についての突っ込んだ話はあまり聞いたことがないような
  - いろいろトラブってるのはうちだけかな？
- まあ、キャッシュサーバーについて、いろいろネタが溜まったので、ぶちまけてみましょう
  - authoritative sever にもいろいろ面白い話はあるけど、またの機会に

キャッシュサーバーとは

- クライアントからの再帰問い合わせに答える
  - 他のネームサーバーへ問い合わせ、得られた答えをキャッシュ
  - トラブると、「インターネットが使えません！」
- ゾーン情報を持たない
  - 最近はゾーンを持つ権威サーバー(authoritative server)と分離する
  - 例外もある(小規模LAN内のDNS、RFC1918逆引きゾーンなど)
- 非常に重要なのに、あまり話題にならないのはなぜ？
  - 設定に難しいところもない？
  - 問題もあまり起こってないのかな？
- トラブルに気がついてないだけかも？

キャッシュサーバーをいろいろ調べてみたきっかけ

- もともと BIND8 を使っていた
- BIND9 を試してみよう
  - view を使って遊んでみたい
  - BIND8 はセキュリティホールが多い
- 結果…実用に至るまで、長い道のりが
  - 主にパフォーマンス問題
    - よく言われていますが、実際のところ？
    - 一口に「パフォーマンス」というけど？
- 将来的なことも考え、ハイパフォーマンスな商用ソフトウェアも評価してみる
  - BIND + ロードバランサという選択肢もある
    - 今回は試さなかった（ロードバランサはあまり好きではないため）

BINDの各バージョンの比較(BIND8)

- BIND8
  - 少し前まで主流だったと思われるバージョン
  - 昨年、各種セキュリティホールがあったせいで、BIND9にだいぶ移行が進んだかも
- BIND8.2.x (latest は8.2.7)
  - 現在広く使われているBIND8の原型
  - セキュリティホールが修正されていない
- BIND8.3.x (latest は8.3.7)
  - EDNS0 に対応
  - セキュリティホール的には問題ないはずだが…
- BIND8.4.x (latest は8.4.4)
  - IPv6 transport に対応
  - 実はそれ以外にもいろいろ修正されている (詳細は後述)

BINDの各バージョンの比較(BIND9)

- BIND9
  - ISC との契約により、Nominum のエンジニアが実装
  - BIND8 とはまったく別にスクラッチから実装
  - セキュリティ、ポータビリティ、マルチプロセッサスケーラビリティ
- BIND9.2.x (latest は 9.2.3)
  - BIND9 リリースの最新版
  - 9.2.4rc6 が 7/6 に出ているので、まもなく 9.2.4 か
- BIND9.3.0
  - 7/6 に 9.3.0rc2、まもなくリリースか
  - rrset-order、check-names 等、BIND8 と機能的に同等に
  - 他にも rndc flushname など、細かいところでお勧め

キャッシュサーバーのパフォーマンス

パフォーマンスの定義、計測方法、比較

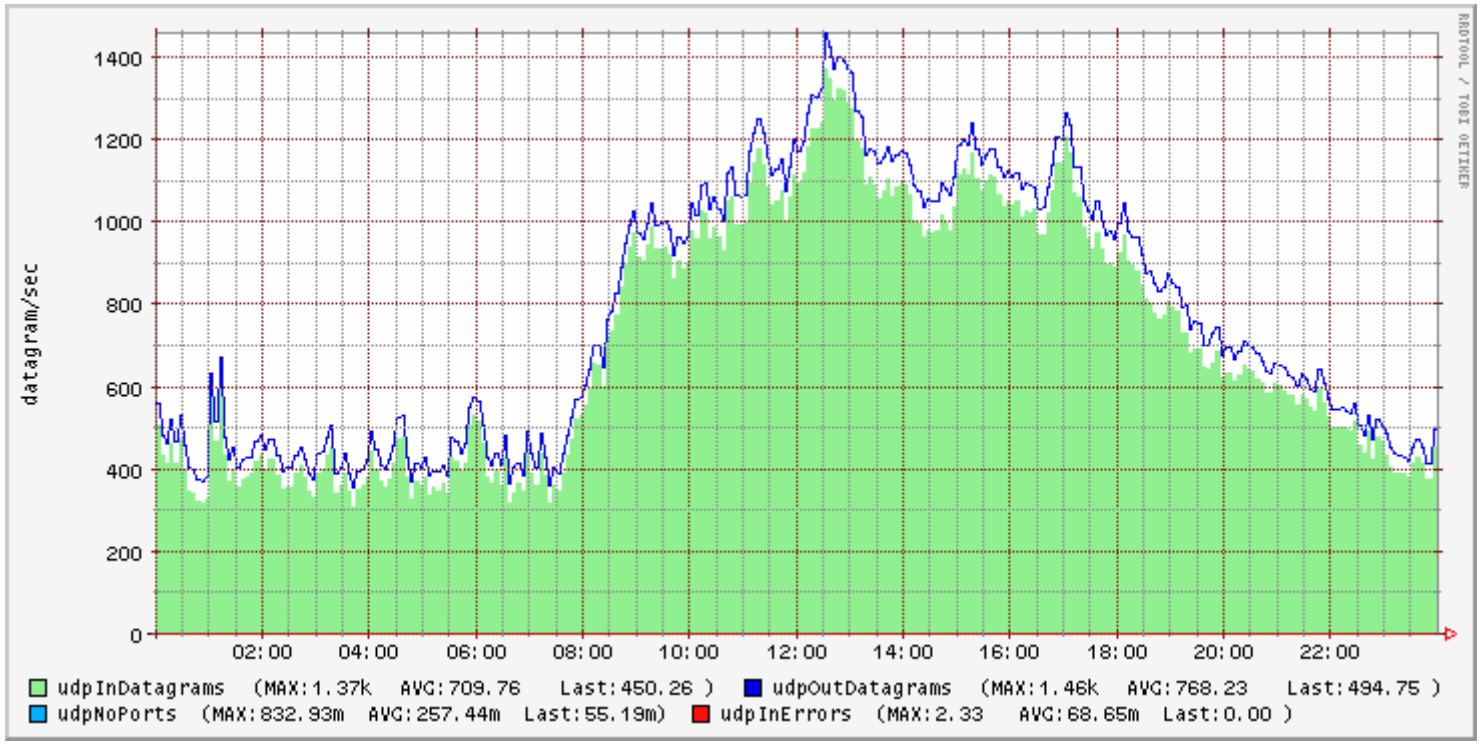
キャッシュサーバーのパフォーマンスとは

- よくあるのは「何qps」という表現
  - query per second
  - 一秒間に何 query「さばけるか」という数字
  - おそらく、query をこぼさずに応答できる上限の query 数
- どういう条件で計った数値？
  - キャッシュの状態、ヒット率
  - 計測方法
- qps だけでいいの？
  - latency もわりと重要だったり

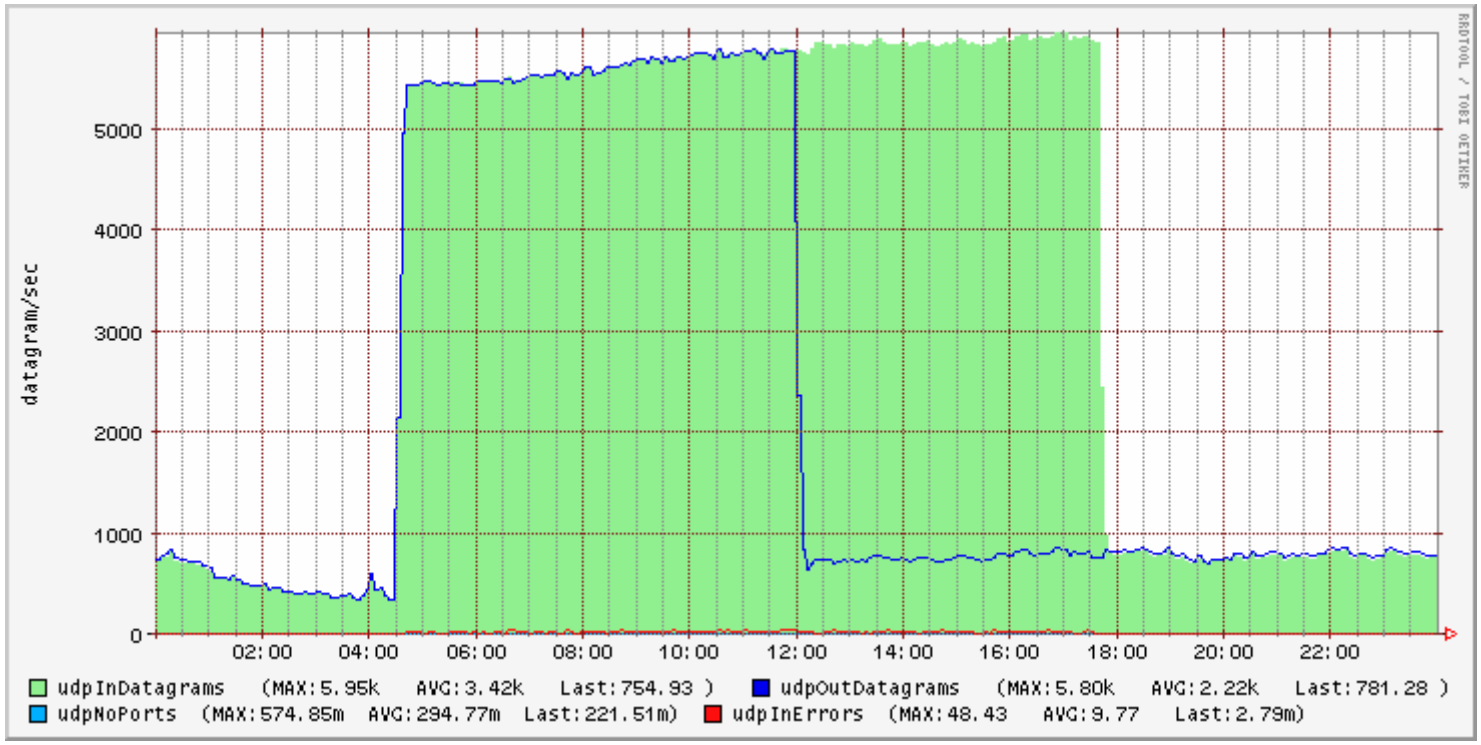
qps の計測・把握

- パフォーマンス(qps)の測定の前に…
- 普段自分の管理している DNS server にどれぐらいの query 数・qps が来ているか、把握していますか？
- 計測方法の例：BINDの統計情報を使う
  - 5分ごとに ndc/rndc stats した結果をスクリプトで処理して、SNMP等で拾う
  - リアルタイムさにかけるが、あまり頻繁に ndc stats したくはない
- udp パケット数を計測してみる
  - DNS サーバー専用マシンなら、udp packet 数　DNSのパケット数
  - DNSのパケット数は、ほぼ query 数に比例する(1〜2割増し程度?)
  - near realtime に傾向を掴むのによい
  - カーネル内で計測されており、SNMPで容易に収集可能

udp packet 数の計測



udp packet 数の計測(異常 query を観測)



udp packet 数の計測(異常 query を観測)

- 単一 IP address から 5000qps 以上のレートで query
  - 12時頃に気がついて、src IP address を blackhole
- 10000qps ぐらいの(ほとんど DoS のような) query が来たことも
  - この時は、動作のおかしくなったブロードバンドルーターが原因
  - 最近多いのは、マスメーリング型ワームからの大量 query
    - SERVFAIL となるような query の場合、ひたすら query を出しまくる
- よくグラフを観察してみると
  - 異常 query が来ている時は、udpInErrors が観測されている

query drop / packet drop

- udpInErrors
  - **プロセスが処理せずに、カーネル**(socket buffer)**内で捨てられた** udp packet **の数**
  - named **が処理しきれないレートでパケットが到着していることを示している**
  - query drop / packet drop
- **定常的に** udp packet **数を計測してみると**
  - **ピークで** query drop / packet drop **をしているサーバーが観測される**
  - **比較的古いハードを使った非力なマシン**
  - **いい機会なので、マシンをリプレイスして、強化**
    - query drop / packet drop **は観測されなくなる**
  - 1% drop **程度だと、監視に引っかからないことも…**
- **気がつかないだけで、異常なレートで** query **が来ていることは結構ある**

パフォーマンス(qps 処理能力)を把握しておこう

- 通常時、どの程度の qps まで処理できるのか、計測して把握しておくことは、キャパシティプランニング上重要
  - query 数の増加傾向をみて、マシン増強の計画を立てる
- 異常時もどの程度まで耐えるのか、数字があると安心できる
  - ただし、SERVFAIL になるような query は、通常の query に比べて辛いことが経験上わかっている
- この際だから、いろいろなキャッシュサーバーのパフォーマンスを計測してみる
  - BIND8に比べてパフォーマンスが劣ると言われる BIND9 はどの程度劣るのか
  - BINDの数倍のパフォーマンスと言われる Nominum CNS も試してみたい
  - ついでなので djbdns(dnscache) も

性能測定のための実験環境

- **ホストを 2台準備する**
  - server と query
  - **2台は同一セグメントにいる**
  - **2台の間の** round-trip time **は** 0.2 msec
- server
  - **ネームサーバが動作するホスト**
- query
  - query **を投げて測定するホスト**

実験環境 その2

	server	query
cpu	Pentium III 850MHz	Pentium 4 2.20GHz
memory	768Mbyte	1024Mbyte
HDD	SCSI 18Gbyte	SCSI 18Gbyte
OS	FreeBSD 4.9_RELENG	BSD/OS 4.3.1
sysconfig	net.inet.udp.recvspace= 319000 kern.ipc.maxsockbuf= 2557000	net.inet.udp.recvspace = 319000 net.socket.sbmax = 2557000

計測対象ソフトウェア

- BIND8.3.7
  - BIND8.3系の最新版
- BIND8.4.4
  - CHANGES をよく読むと、IPv6 対応以外にも多くの修正があるので対象とする
- BIND9.2.3
  - BIND9.2系の最新版
- Nominum CNS 1.3
  - ハイパフォーマンスが売りの商用DNSサーバーソフトウェア
  - 現在は1.4がリリースされている
    - パフォーマンス面でも若干の改善があるらしい
- dnscache 1.5 (djbdns)
  - 基本的には使う予定はないが、よい機会なので比較してみる

スループット測定の方法

- Nominum 版 queryperf を利用

  ftp://ftp.nominum.com/pub/nominum/queryperf-nominum-2.0.tar.gz
  - -q オプション(同時に送信できるクエリの数)を利用
    多くのクライアントからアクセスを受けた状態を疑似的に再現
  - server ホストで CPU 利用率を見て IDLE が生じていないことを確認
  - バッファサイズを大きく
  - 同時クエリ数 500, timeout 5秒, 100秒間測定

  queryperf -s $SERVER -d $LIST -b 2497 -q 500 -t 5 -l 100

- リストは実サーバのログから抽出したもの

- キャッシュされている状態で測定

http://www.nominum.org/content/documents/CNS_WP.pdf 参照

latency の測定方法

- Nominum 版 queryperf を利用
  - latency をヒストグラムにして表示することができる
  - Nominum 版固有の機能
  - 1msec 間隔で latency を測定
  - バッファサイズを大きく
  - 同時クエリ数 500, timeout 5秒, 100秒間測定

queryperf -s $SERVER -d $FILE -b 2497 -q 500 -t 5 -l 100 –H 1000

- リストは実サーバのログから抽出したもの
- キャッシュされている状態で測定

ネームサーバのコンフィグ

- dnscache

cat dnscache/env/CACHESIZE
200000000

cat dnscache/env/DATALIMIT
300000000

cat dnscache/env/IP
query のアドレス

ネームサーバのコンフィグ

- BIND8.3.7,BIND8.4.4,BIND9.2.3

```
options {
 directory "$named_rootdir";
};
zone "." in {
 type hint;
 file "root.cache";
};
```

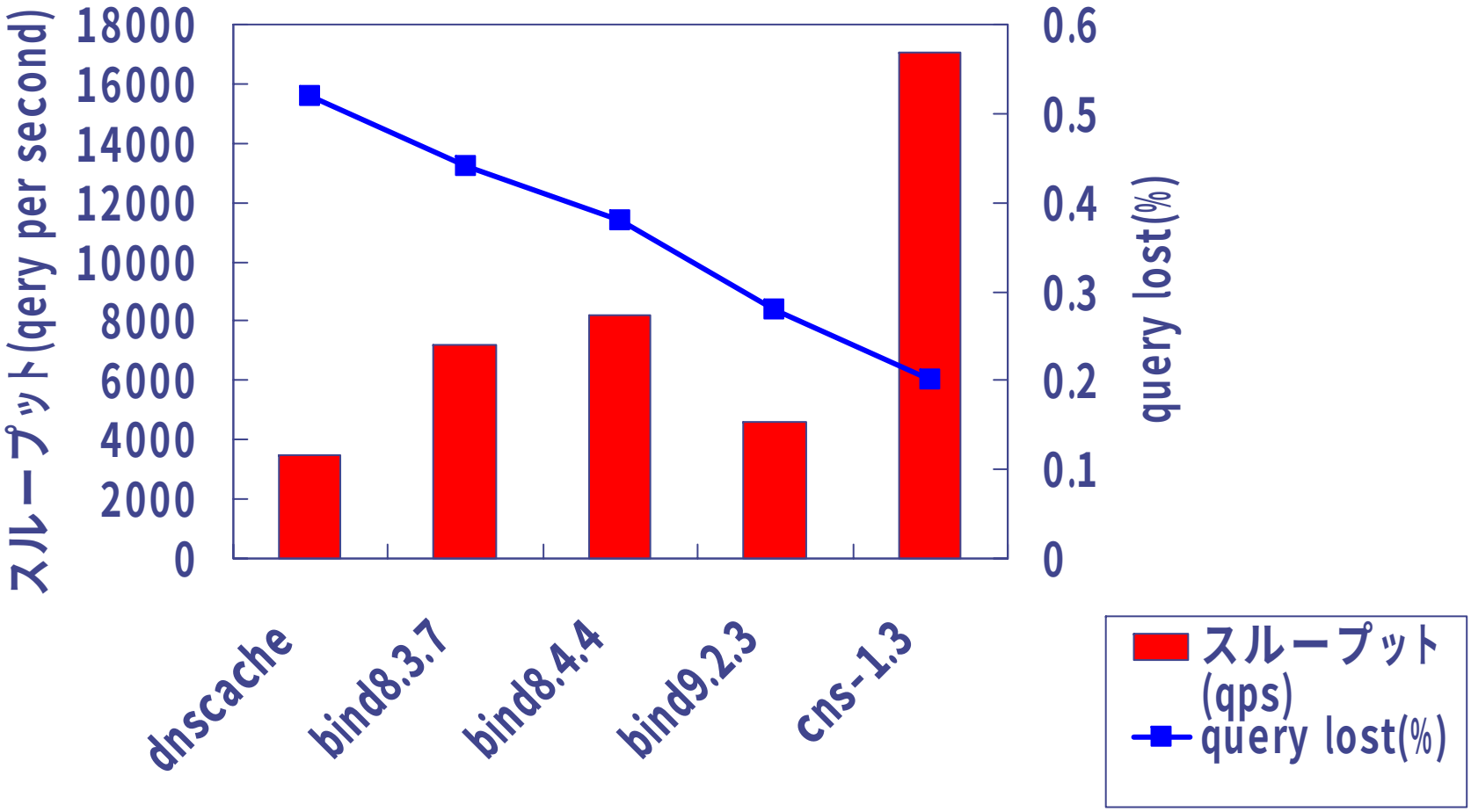
(BIND9 **のみ** rnd **用の設定**)

ネームサーバのコンフィグ

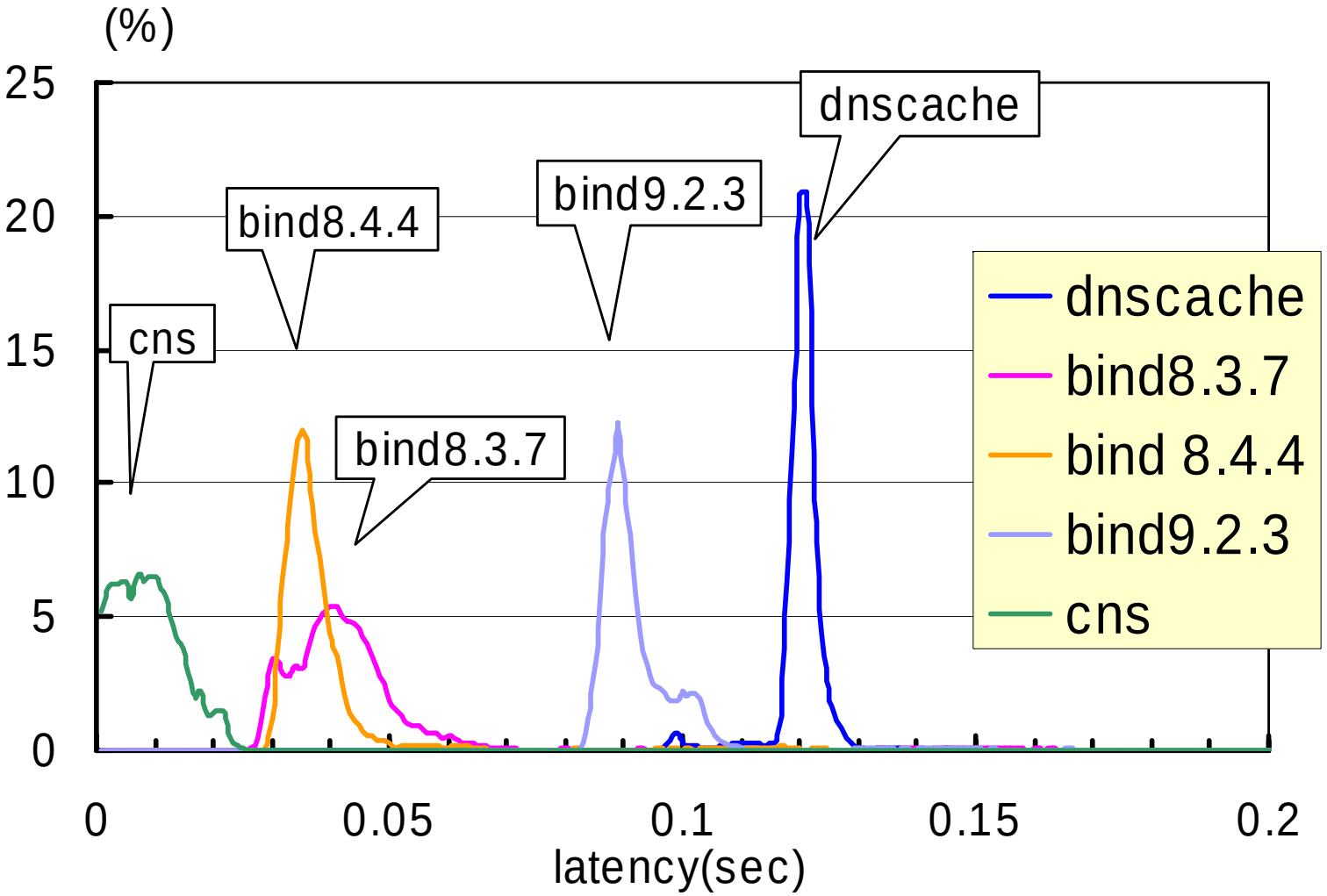
- CNS

```
listen-on 0.0.0.0;
    max-cache-size 200M;
    view "world" IN {
            preload 1.0.0.127.in-addr.arpa. PTR localhost;
            preload localhost. A 127.0.0.1;
    };
```

スループット(qps)と query lost rate



latency



結果

- よく言われているように、BIND9はBIND8の半分のスループット(qps)
  - latency でも明確に差が出た
  - 意外に、BIND8.3.7とBIND8.4.4の差が結構ある
  - BIND9 は、スループットの割りに、query loss が少ない
- Nominum CNS が宣伝文句どおり、ダントツのパフォーマンス
  - スループット(qps)だけでなく、latency も非常に優秀
- dnscache がちょっと奮わないが…
  - 特に latency が高め

dnscache のパフォーマンスについて – logging の影響

- log を処理するプログラム(multilog)が CPU を食っている
- log をすべて捨てるように設定したけれどそれでも 15%程度は CPU利用率を消費している
- クエリ毎に 3行ほどログを吐く
- config で吐くログを少なくはできない
  - 使い込んでいないのでよくわかってないだけかも
  - どなたかご存知でしたら教えてください
- 他のソフトウェアでも querylog をすべて取る設定で計測してみても面白いかも
  - 今回は時間もあまりなく試していない
  - 現実の環境ではすべての querylog を取ることはあまりない

パフォーマンス計測についての考察

- BIND9 も、それなりに使えないことはないかも
  - ただし、BIND8ほど異常時への余裕はない
    - DoS とか考えるとちょっと怖いかな
  - でも BIND8 のセキュリティホール対策には疲れたし…
  - キャッシュサーバーの挙動としては、BIND9 のほうが好ましい
- Nominum CNS が使えればベストでしょう
  - ただし、結構お値段が…
    - 住商エレクトロニクスの人をつかまえて聞いてみてください☺
  - 「複数台のBIND9 + ロードバランサ」となら十分比較対象となりうる

BIND9 を実戦投入してみると...

そろそろBIND9を実戦投入してみたいな

- セキュリティホールも多いし、BIND8 とはそろそろお別れしたい
  - キャッシュサーバーとしての挙動的にもBIND9のほうが望ましい
- とあるプロジェクトで、キャッシュサーバーで view を使うようなアイデアを試してみたいという声が
- というわけで、BIND9 をキャッシュサーバーに投入してみると…

BIND9 の「サクサク感が無い」問題

- BIND9 をキャッシュサーバーとして使っていると、web ブラウジングの際目的のページが表示されるまでに一瞬待たされる感じがする
- このことを「サクサク感が無い」と呼ぶ(2chより)
  - 目的のページがさくさくと表示されない、ということか
- 過去に 2回ほど、キャッシュサーバーへの BIND9 の投入をトライして断念して経緯が
  - 投入してみたものの、「サクサク感がない」ということできり戻し

「サクサク感がない」問題のいくつかの原因

- まず最初、log の出しすぎが疑われた
  - 細かなログが取れるようになったことに喜んで、ついいろんなログを取ってみる設定をした
  - 多少、トリッキーなことをやっていたので、それもログをいろいろ取りたかった原因
  - プロファイリングをしてみたところ、ログの出力に足が引っ張られているようだ
  - log のとり方を工夫してみたが、多少は改善するものの思ったほどではない
    - syslog 利用
    - とるログを厳選
- 次に EDNS0 クエリが疑われた
  - tcpdump をしてみると、BIND9 は EDNS0 クエリを行っていた

EDNS0 とは

- DNS は udp packet のサイズの上限を 512bytes としている
- 今後、この 512bytes 制限は問題になるかも
  - IPv6 への移行 (Aレコードに加え、AAAAレコードも)
  - DNSSEC
  - MARID その他いろいろ
- これを拡張するための仕組みが EDNS0
  - RFC2671
  - OPT RR とそれに対する返答によってバッファサイズを通知

EDNS0 対応問題

- BIND9 は、まず EDNS0 で query を投げる
  - FORMERR や NOTIMPL 等が返ってきて、相手のサーバーが EDNS0 に対応していないことがわかると、通常の query で聞きなおす
  - 以後 そのサーバーに対して EDNS0 では query しない
  - 特定のサーバーに対して、EDNS0 で問い合わせをしないという設定はできる
  - 全体として EDNS0 を抑制する方法はない
- 相手のサーバーが EDNS0 に対応していない場合、少なくとも1往復分遅延が発生する
  - FORMERR や NOTIMPL が返る場合
  - 相手がだんまりになる場合、タイムアウトするまで待たされる

EDNS0 は無実

- 確かに初回問い合わせで 1RTT 分遅延するが
  - 国内であればたかだか数十msec
    - 海外(RTT150msec〜)であれば多少は影響あるかも
  - 2回目以降の問い合わせには影響しない
- BIND8.3 で同様に EDNS0 が実装されたが「サクサク感がない」問題はおこっていない
  - 逆に、EDNS0 に対応していない BIND8.2.7のセキュリティホールが修正されない（BIND8.3系以上にアップグレードするしかない）ことから、世間の EDNS0 対応が進んだ
    - ENDS0 の deployment を進めようという ISC の強い意思を感じる

「サクサク感がない」問題の本当の原因

- IPv6 コネクティビティがないのに、IPv6 で query しようとしてタイムアウトしていた！
- 問題の発生条件
  - IPv6 enableでBIND9 がコンパイルされている
    - BSD/OS や FreeBSD ではカーネルIPv6 対応であることにより、configure で自動的に検出されて enable-ipv6=yes となる
  - IPv6で外部のホストへの到達性がない
  - 問い合わせ対象のNSサーバがA,AAAA両方持っている
    - .JP の NS サーバー群には AAAA がついている！
  - NSサーバの AAAA をキャッシュした

「サクサク感がない」問題の本当の原因

- 現象
  - IPv6のアドレスを用いて(AAAA のアドレスに対して) query を行おうとして doio_send が ”No Route to Host” で失敗し、IPv4により(A のアドレスに対して)リトライが行われる
  - doio_sendは即時失敗しているにも関わらずリトライまでのタイムアウト 1秒待つ
- 対策
  - 明示的に --disable--ipv6 として configure する
    - 最近はカーネルは IPv6 対応している OS がほとんどのはず
  - IPv6 の到達性を持たせる☺
  - bind-users に流れていた神明さんパッチをあててみる
    - 2004-06-23 のメール
- 日本以外では気がつきにくい問題だったかも
  - 日本では *.dns.jp に AAAA がついているため問題に遭遇しやすい

BIND8 のトラブル

BIND8.3 のキャッシュサーバで
名前が引けなくなったことありませんか?

BIND9 はしばらく諦めて、BIND8を使い続けましょう...

- 「サクサク感がない」問題が解決したのは、わりと最近の話
  - たしか今年の5月ぐらい
- それまでは、「まだまだBIND8を使うしかないよね」という話に落ち着いていた
- ところが、ある時期から、BIND8 のキャッシュサーバーで引けないドメインがたまにでてくるようになった
  - 顧客や社内からの問い合わせ
  - 致命的な場合は、named の restart で対応するしかない
- 試しに BIND9 のキャッシュサーバーで引いてみると引ける
- ただし、そのようなケースでは、そのドメインの設定に問題があるケースが多い
  - NSのうち片方が、いわゆる lame delegation となっている

引けない問題を詳しく調べてみると - BIND8.3.7 で陥る罠

- zone の authoritative name server の片方が lame でかつその zone に存在しないという条件下
- TTL の関係等の状況によって lame でない方の name server の A のみ expire したとき
  - キャッシュの状態

example.com 1000 NS ns0.example.com
example.com 1000 NS ns1.hoge.net <- lame
ns1.hoge.net 100 A 192.168.1.1
(ns0.example.com A 192.168.0.1) <- expired

- example.com の NS が expire するまで、キャッシュされていない example.com ドメインのあらゆる名前が解決できなくなる

何故!?

BIND8.3 系だけ起こるようですが...
ちょっと詳しく調べてみました

手がかり: 複数の NS からの選び方

- 一般に RTT の小さい name server に問い合わせをするといわれている
- BIND8.3.7
  - ./bin/named/ns_forw.c
    - qcomp
  - lame なものは優先度を下げる
  - ある程度 RTT の差が大きければ RTT の小さいものを優先
  - トポロジー的に近いものを優先
    - local network or topology, sortlist で指定
  - RTT の小さいものを優先
- BIND9.2.3
  - RTT しか見ていない様だ
  - topology 設定はまだ実装されていないらしい

BIND 内部で保持している RTT の値の更新の仕方

- BIND8系 BIND9系(9.2.2以降) 共通
- その name server に問い合わせたとき
  - new_rtt=old_rtt * 0.7 + rtt * 0.3
- その name server に問い合わせなかったとき
  - new_rtt=old_rtt * 0.98
- たまには RTT の大きな name server にも聞きにいくことによって，変化(障害等)に対応
  - RTT の小さい name server がたまたま落ちていた場合にも，その server が復活したらそちらにクエリを投げるようになる

ちょっと脱線 - dumpdb で BIND の持っているRTT情報 を確認してみる

- BIND8 の dumpdb
  - NT=の部分に RTT の値が出力される
  - dumpdb の上の方にコメントされている

```
; Note: Cr=(auth,answer,addtnl,cache) tag only shown for non-auth RR's
; Note: NT=milliseconds for any A RR which we've used as a nameserver
```

  - src/bin/named/db_dump.c 346 行目
    dp->d_nstime を NT の値として出力している
    src/bin/named/db_defs.h

```
u_int16_t    d_nstime;    /* NS response time, milliseconds */
```

BIND8 の dumpdb


```
$ORIGIN ad.jp.
iij     86382   IN      NS      dns0.iij.ad.jp. ;Cr=auth [210.138.175.5]
        86382   IN      NS      dns1.iij.ad.jp. ;Cr=auth [210.138.175.5]
$ORIGIN iij.ad.jp.
dns0    86382   IN      A       210.138.174.16  ;NT=24 Cr=addtnl [165.76.0.98]
www     1782    IN      A       202.232.2.10    ;Cr=auth [210.138.175.5]
dns1    86382   IN      A       210.138.175.5   ;NT=3 Cr=addtnl [165.76.0.98]
```

BIND9 の dumpdb


```
; authauthority
iij.ad.jp.          86398   NS      dns0.iij.ad.jp.
                    86398   NS      dns1.iij.ad.jp.
; glue
dns0.iij.ad.jp.     86398   A       210.138.174.16
; glue
dns1.iij.ad.jp.     86398   A       210.138.175.5
; authanswer
www.iij.ad.jp.      1798    A       202.232.2.10
```

BIND9 の dumpdb で RTT を見たい

- lib/dns/view.c

```
1167 #ifdef notyet /* clean up adb dump format first */
1168         dns_adb_dump(view->adb, fp);
1169 #endif
```

  - 上記の dns_adb_dump を有効にしてあげると RTT も dump される

- しかし lib/dns/adb.c によれば

```
2935    /*
2936     * Lock the adb itself, lock all the name buckets, then lock all
2937     * the entry buckets.  This should put the adb into a state where
2938     * nothing can change, so we can iterate through everything and
2939     * print at our leisure.
2940     */
```

- 実運用では利用しない方が良いかも

BIND9 の dumpdb その2


```
; dns0.iij.ad.jp [v4 TTL 4] [v4 success] [v6 unexpected]
;       210.138.174.16 [srtt 24]
; dns1.iij.ad.jp [v4 TTL 4] [v4 success] [v6 unexpected]
;       210.138.175.5 [srtt 2620]
```

BIND8 での名前解決の流れ

- nlookup() で DB にキャッシュされているかを探す
- findns() でキャッシュにある or 知っている NS を探す
- ns_forw() で得られた name server に基づいて名前解決を試みる
  - qnew() 内 find_zone() を利用して forward とか stub の zone を探す
  - nslookup() を利用して nameserver のアドレスを取得
  - どの NS がよいか sort qcomp()
  - query 送る

原因

- nslookup() では expire した ns0.example.com の IP アドレスを取得できない

- nslookup() で取得できる example.com のネームサーバの IP アドレスは 192.168.1.1 (lame な方のネームサーバ)のみ

example.com 1000 NS ns0.example.com
example.com 1000 NS ns1.hoge.net <- lame
ns1.hoge.net 100 A 192.168.1.1
(ns0.example.com A 192.168.0.1) <- expired

BIND8 8.2, 8.3, 8.4 系での違い

- BIND8.2.7
  - nslookup() で lame な NS を除いている
- BIND8.3.7 (BIND8.3.4 -> BIND8.3.5 でエンバグ)
  - nslookup() で lame な NS を除かない
  - qcomp() で lame かどうかに基づいて NS の優先度を決めている
  - nslookup() で glue レコードが取得できない
- BIND8.4.4 (BIND8.4.3 -> BIND8.4.4 で修正された模様)
  - nslookup() で glue レコードも取得できるようになった
    - CHANGES

      1637. [bug] if the current lookup requires self glue allow nslookup to signal that the caller may call check the parent.
    - src/bin/named/ns_forw.c 738-741

```
/*
 * Allow nslookup to tell the caller to go up one level
 * to look for glue.
 */
```

結論・感想

- 現状は、BIND8.4.4にアップグレードするしかない
  - IPv6 対応のコードとか、かなり変更点が多くて心配だけど…
- 結構重大な問題だと思われるが、ISC からははっきりとしたアナウンスがない
- いろいろ情報を調べたが、bind-users は(英語の上) S/N 比が低くて有益な情報を拾い上げるのが大変
  - 結局、今回の問題などはソースを追って自前で解決

IIJ
Internet Initiative Japan

Network Initiative

BIND9 のトラブル again

なんかクエリこぼしてるみたいなんですけど…

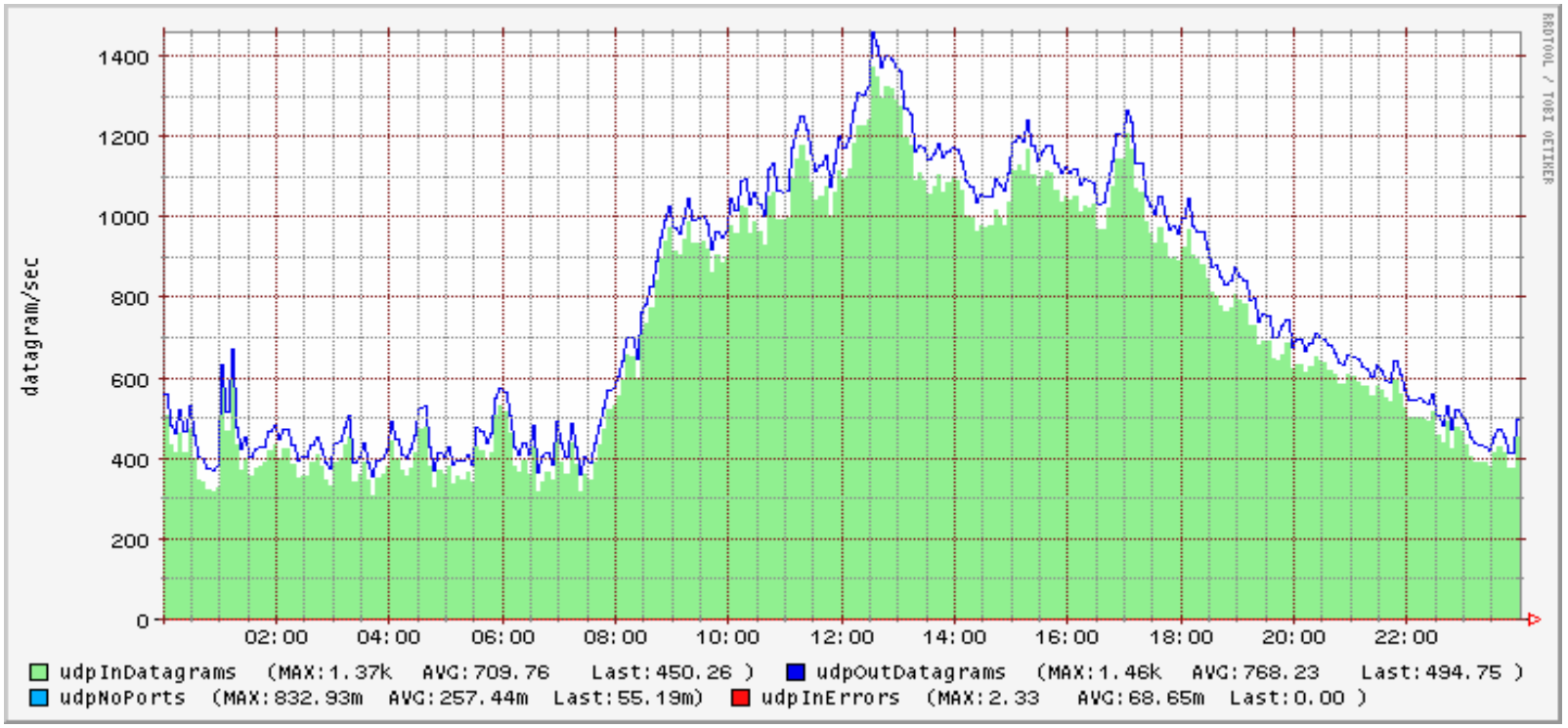
さて、「サクサク感がない」問題も解決したし

- 「サクサク感がない」問題も解決したし、BIND8 のトラブルもあるので
  - 8.4.4 で直ってることが判明したのは、実は先週ぐらい
- 徐々に BIND9 のキャッシュサーバーも実用化してみましょうか
  - パフォーマンスの余力が心配なので、まずは、顧客に提供するキャッシュサーバーではなく、社内で運用しているサーバーから参照するキャッシュサーバーから試してみる
- いい感じでサクサク使えてるようだ
  - 調子にのって 9.2.3 に、9.3.0 beta から rndc flushname をバックポートしてみたり
    - 特定のキャッシュの内容を消せるので、実運用では非常に便利です

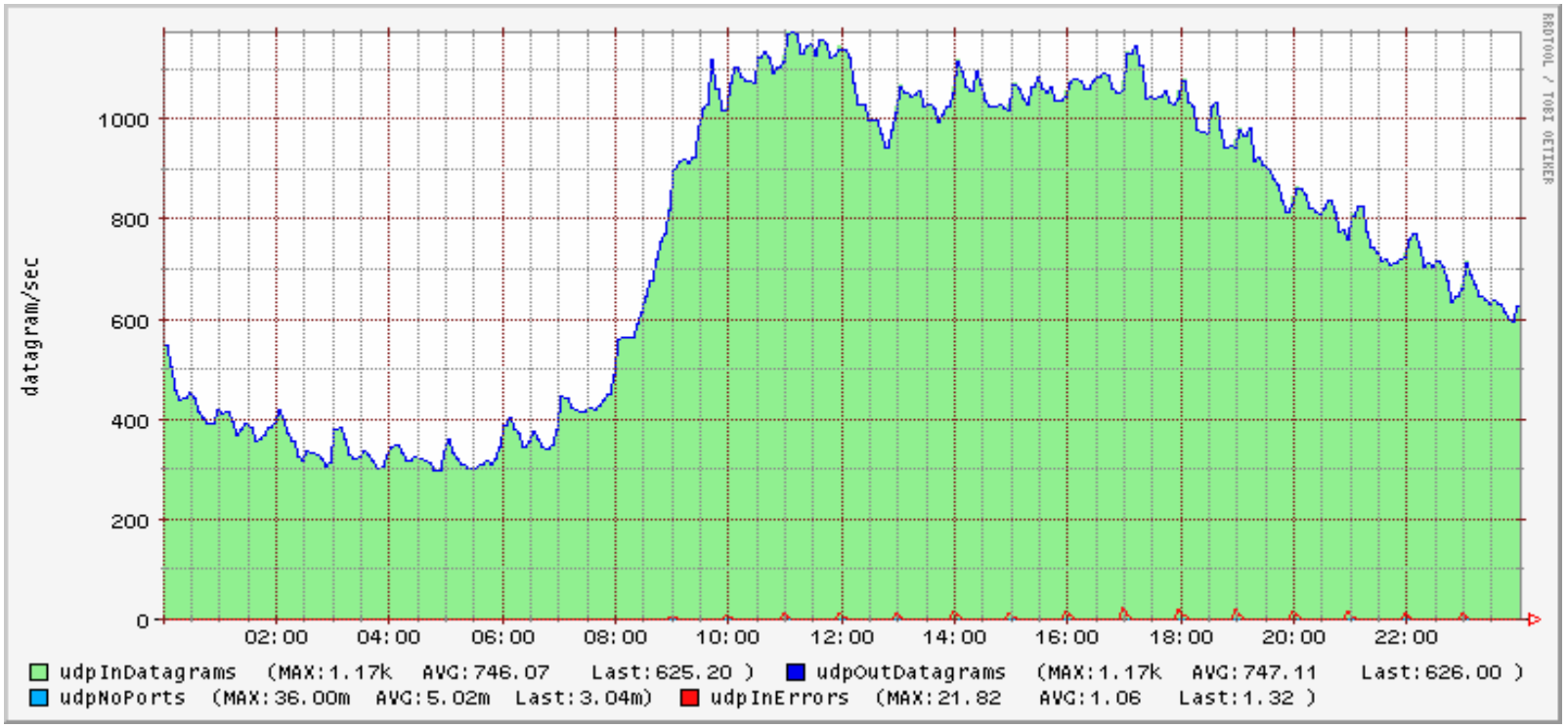
ん？

- udp packet **数のグラフ見ると、なんかドロップが**…
- query **数がある程度以上の時間帯は、1時間ごとに周期的にドロップが起きているようだ**

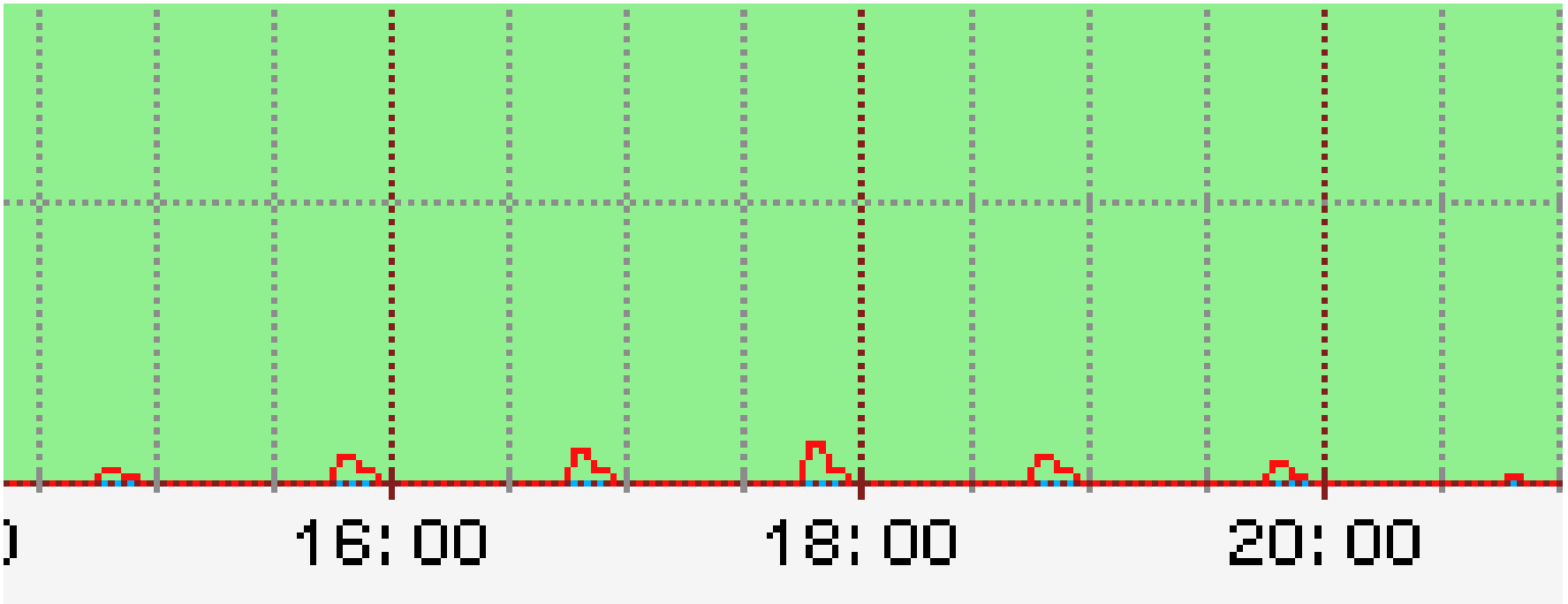
udp packet 数グラフ 正常なもの



udp packet 数グラフ 1時間おきにこぼしている



udp packet 数グラフ 1時間おきにこぼしている様子 拡大



クエリをこぼしている原因

- 1時間ごと、というのはなんかとっても怪しい
  - なんか内部でメンテナンスタスクが走ってるんじゃない？
  - man とかバッタ本とにらめっこして、設定をいろいろ調べてみる
- いろいろ調べてみると、cache の cleaning のタイミングでこぼしているようだ
  - cleaning-interval は default で 60分
  - cleaning-interval 0 にするとこぼさなくなった！
    - しかし、当然のようにどんどんメモリを浪費し続けるのでよろしくない
  - cleaning-interval を長くしたり短くしてみたりしたが効果なし
- マシンを PentiumIII 850MHz から Pentium4 2.4GHz にアップグレードしてみてもあまり効果なし

クエリをこぼしている原因 – ソースを追ってみると...

- 根本的な原因は BIND9 の構造的問題のようだ
  - マルチプロセッサ環境でのスケーラビリティのために、コードがマルチスレッドを前提として設計されている
- シングルスレッド環境だと厳しい
  - マルチスレッド前提のコードをシングルスレッド環境で動かすためのタスクスイッチの処理が良くない．無理にタスクスケジューリングしている
- cache の cleaning をするタスクが1000エントリ処理するまでタスクのスイッチがおこらないのでその間に多数のクエリがくるとこぼす

対策

- 1000で多いなら減らしてみよう
  - 一度に処理するエントリ数を1000から300にしたところ負荷は減った
    - けどまだちょっとこぼす
  - 結局、うちの環境だと128ぐらいまで減らすとこぼさなくなった
  - lib/dns/cache.c

    45 行目

    ```
    #define DNS_CACHE_CLEANERINCREMENT  1000  /* Number of nodes. */
    ```

    483 行目

    ```
    cleaner->increment = DNS_CACHE_CLEANERINCREMENT;
    ```

- 環境によって違うので、named.conf で設定できるといいよね☺
  - 血気盛んな若者がハックしてくれました
  - 整理して近日中に ISC に報告してみる予定

まとめ

- OS の udp packet 数を記録するのは有益
  - クエリ数の傾向・burst の把握
  - packet drop の発見
    - ndc stats ではわからない
- BIND はセキュリティ対策でなくても最新版を使ったほうがよさそう
  - 8/9 各バージョンでそれぞれ言える
  - CHANGES にはさらっとしか書いてないが、重要な修正をしていることが
- BIND9 実用のためにはそれなりに大変
  - サクサク感問題
  - クエリこぼしたり
  - 潜在的な問題が他にもあるかも
  - まあそろそろこなれてきたかな

TODO 今回時間がなくてやり残したこと

- BIND8.3.7 と BIND8.4.4 のパフォーマンスの差の原因は？
- BIND9 のマルチスレッド環境でのパフォーマンス検証
  - シングルCPUでスレッドあり・なしでパフォーマンス比較
  - シングルCPUでスレッドあり・なしでcleaning時のパケットドロップ比較
- BIND9のマルチプロセッサ環境でのパフォーマンス検証
  - シングルCPUとマルチCPUでのパフォーマンス比較

Special Thanks To...

- **IIJシステム開発部 阿久津君**
  - **パフォーマンス測定、Nominum CNS の評価を手伝ってもらいました**
- **IIJシステム技術部 小林君**
  - **BIND9のIPv6問題解明**
  - **BIND9のcleaning-interval でquery をこぼす問題の対策**
  - **rndc flushname のBIND9.2.3への移植**
  - **他いろいろ**
- **IIJネットワーク技術部の松崎君**
- **IIJシステム技術部のみなさん**

- **JANOG14 PC の丸山さん、佐々木さん、廣海さん**
  - **土壇場まで資料ができずにやきもきさせ、ご迷惑をおかけしました_o_**

おしまい

ご静聴ありがとうございました