DNSの評価と計測の話

Internet Week 2013

SCSK株式会社
服部　成浩

内容

- DNSストレスツール Nominum dnsperf/resperfの紹介
  - dnsperf と resperf の違い
    - 負荷の生成方法
  - エラーメッセージ対処

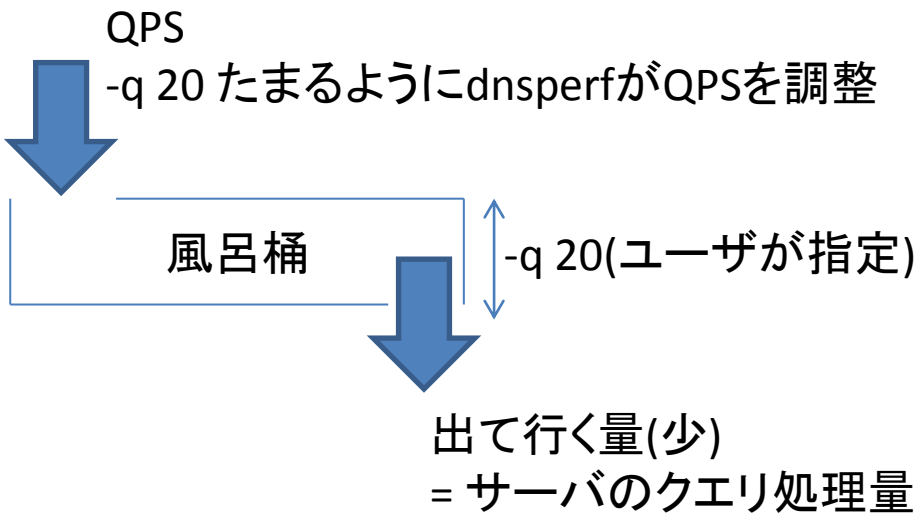
- 同じDNSサーバ(unbound)に対して、dnsperf, resperfでかけた場合、同じ結果になる?
  - ケーススタディ
    - dnsperf, resperf オプション指定例、結果ファイル

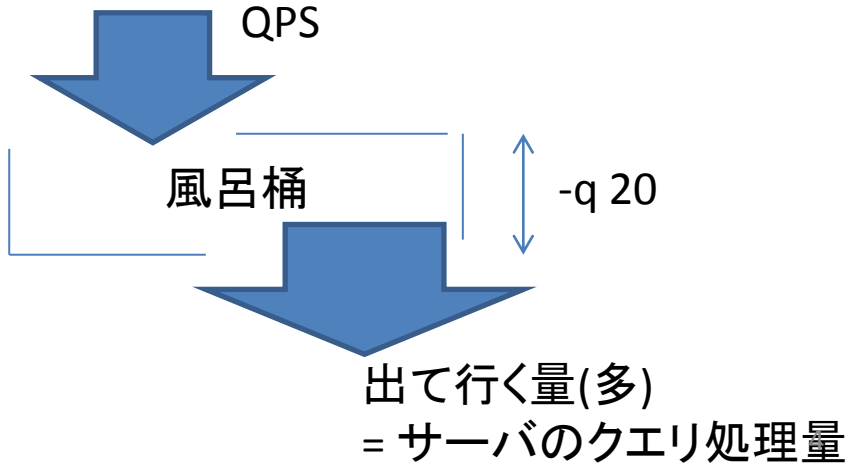
Nominum dnsperf/resperf

- dnsperf
  - 権威サーバや、LAN環境でのキャッシュサーバのテストではO.K.
  - キャッシュのテストでWAN回線を使用した場合は結果が不十分になる可能性あり
    - レスポンス状況でdnsperfの出す負荷(qps)が変化するため
- resperf
  - レスポンス状態にかかわらず負荷を上げていくことができる
    - したがって、テスト時にWAN環境の影響を受けにくい

dnsperfの負荷
self-pacing approach

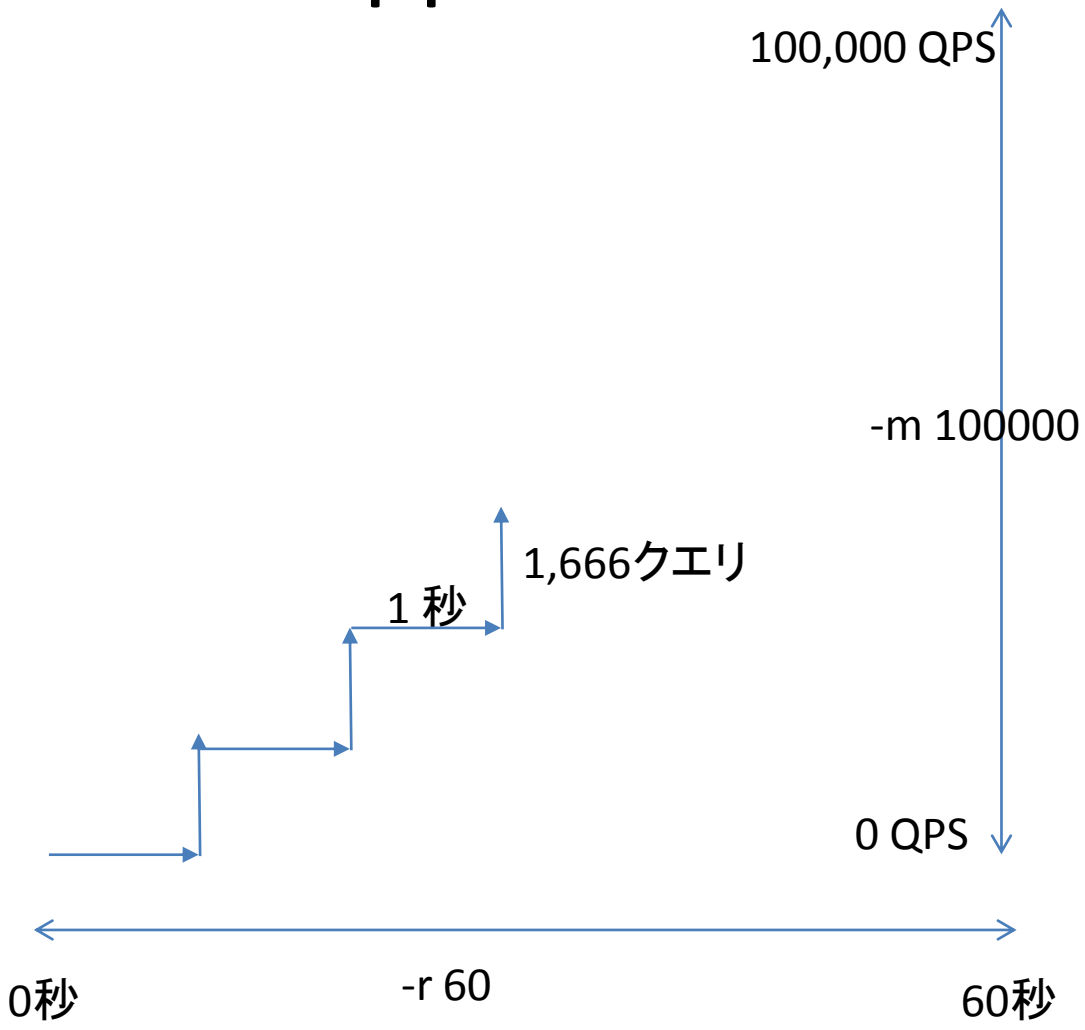
- # dnsperf -q 20の場合
  - 名前解決中のクエリが20たまるようにQPSを調整する
- サーバの処理量(出て行く量)に応じ、QPS(注入する水の量)がかわる
- 遅延のある環境だと、名前解決中のクエリがたまりやすい
  - QPSは少ない





resperfの負荷
controlled rate approach

- 60秒かけて100,000 QPS(デフォルト値)
  - -r 60
  - -m 100000
- 毎秒、約1,666 クエリ、負荷を上げていく



dnsperf
オプション指定例

- # dnsperf -s 192.168.11.21 -S 1 -d query_random_list.txt –q 20 –c 1 –b 2048000
  - -s ：ターゲットのIP
  - -S 1 : 毎秒dnsperfが送信したqpsを表示
  - -d : クエリファイル
  - -q 20 : 名前解決中のクエリが 20 になるように、QPSを調整する (デフォルト –q 100)
  - -c 1 : クライアント数 (デフォルト –c 1)
  - -b : UDPバッファ (デフォルト32 kilo bytes)

dnsperf –q オプション 遅延によるQPS影響

- 遅延なし、遅延100msec の環境で同じ負荷を発生させた場合
- dnsperfのオプションは同じ
  - # dnsperf -s 192.168.11.21 -S 1 -d query_random_list.txt –q 20

- 遅延なし (4,600 QPS)
  - dnsperf上の送信QPS
    - 1383974449.381060: 4584.516918　← エポック時間: QPS
    - 1383974450.382290: 4629.305954

- 遅延100msec　(100 QPS)
  - dnsperf上の送信QPS (100QPS)
    - 1383974318.808760: 99.863985

- 遅延があると、名前解決中のクエリ(-q 20) を 100QPS程度で維持できるため,負荷(QPS)が小さい
  - 遅延のある環境では期待したQPSとならない場合がある

dnsperf
-q オプション

- -q オプションの値を大きくすればQPS多量に送信できる?
  - 例えば –q 5000?
    - -q の値を大きくしてもあまりQPSが出ない
      - » QPSが安定しない
      - » UDPバッファあふれ

- –qのデフォルト値は100
  - -q 100 (デフォルト値)でテスト対象のリソースの限界が確認できた場合は、-q 50, 20, 10, 5 と小さくしていく
  - -q 100でまだリテスト対象のソースがあまっていれば、-q 200, -q 300と大きくする

dnsperf -b オプション UDPバッファ エラー

- UDPバッファエラー、dnsperfマシンの限界
- # dnsperf -s 192.168.11.21 -S 1 -d query_same_list.txt -l 30 -q 5000
  Warning: failed to send packet: Resource temporarily unavailable

UDPバッファのエラーをチェック (dnsperfマシン上で netstat –su)

```
# netstat –su
Udp:
    <snip>
   7552362 packets sent
   SndbufErrors: 18372
```

・-b オプションでバッファサイズを増やすことでエラーがなくなる場合もある

dnsperf –b 204800 -s 192.168.11.21 -S 1 -d query_same_list.txt

デフォルト : OSのデフォルト値(wmem_default, rmem_default)

送信バッファ: /proc/sys/net/core/wmem_default (wmem_max ではない)

受信バッファ: /proc/sys/net/core/rmem_default (rmem_max ではない)

OSのUDPバッファのチューニング (Ubuntu 12.04)

- 送信バッファ
  - # sysctl -w net.core.wmem_default=10485760
  - # sysctl -w net.core.wmem_max=10485760
- 受信バッファ
  - # sysctl -w net.core.rmem_default=10485760
  - # sysctl -w net.core.rmem_max=10485760

dnsperf –c オプション クライアント数

- -c 1 : 1 クライアント数
  - 1クライアントの定義
    - 送信元UDPポート番号が異なる
    - 注意: 送信元IPアドレスは同じ

- 1クライアントで2スレッド使用
  - 送信で1スレッド、受信で1スレッド
    - 2クライアントだと4スレッド

- 指定方法: -c クライアント数
  - クライアント数 2　　./dnsperf -c 2 -d query.txt -s IP
  - クライアント数 10　./dnsperf -c 10 -d query.txt -s IP

- デフォルト(-c を指定しない場合) は クライアント数は1
- -c の値を増やすと、使用するスレッド数が増えるので、QPSが増える場合がある

- resperfはシングルスレッド。1クライアント(送信元IP*1,送信元UDPポート*1)

dnsperf チューニング

- -b, -c, -q オプションを調整し負荷(QPS)を上げる
  - # dnsperf -s 127.0.0.1 -d query.txt -S 1 -l 60 -q 200 -c 4 -b 2048000
    - 1384162059.363745: 38204.153068
    - 1384162060.364771: 39677.291099
  - # dnsperf -s 127.0.0.1 -d query.txt -S 1 -l 60 -q 300 -c 6 -b 2048000
    - 1384162071.842739: 48371.499825
    - 1384162072.846744: 49619.274804

resperf
オプション

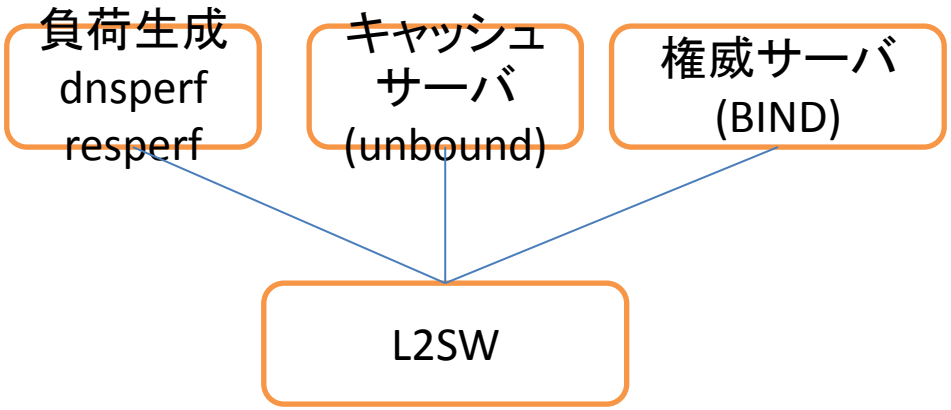
- # resperf -s 192.168.10.10 -d query.txt -r 120 -m 50000 -b 2048000
  - -s : テスト対処のIP
  - -d : クエリファイル
  - -r 120 –m 50000
    - 120秒かけて50,000QPSまで負荷をあげる
    - デフォルトは60秒かけて10万QPS(-r 60 –m 100000)
  - -b UDPバッファサイズ
    - デフォルトは 32 kilo bytes

resperfエラー

- バッファあふれや、リソースの問題でクエリを送信できないときのエラー
  - Warning: failed to send packet: Resource temporarily unavailable
  - Fell behind by 1000 queries, ending test at 25472 qps
- -b でバッファサイズを増やす
  - # resperf –b 204800
  - デフォルト: 32kilo bytes (-b 32768)
- netstat –su でエラーチェック(SndbufErrors)
- バッファサイズを増やしてもエラーが解消しない場合は、OSのUDPバッファのチューニング、負荷発生マシンを複数台にわける、負荷を小さくする、スペックの良いマシンを使用する、など

dnsperf, resperf で同じ結果になる?

- 3台マシンを準備
  - 負荷発生マシン
    - Nominum dnsperf/resperf 2.0
  - キャッシュサーバ (unbound　1.4.16)
  - 権威サーバ (BIND 9.9.4)
- LAN環境(回線遅延なし)
- unbound(single thread)に負荷をかける
  - キャッシュヒット率 0%
- OSは 全て Ubutnu12.04 LTS 64bit
- 低スペックな環境で動作させているので
  パフォーマンス(QPS)自体は問題としない
  - resperf 、dnsperfで同じ結果になるか、
    が目的



dnsperf

- -q の値を変化させる
  - 100,20,10,5
  - -q 100 のときのunboundのリソースが100%だったので、-qを小さい値にする
- QPSがだいたい落ち着くところをさがす
- 限界値
  - 4,500QPS ～ 4,900QPSあたり

dnsperf -q	QPS (dnsperf, unbound上で観測)	unboundのCPU使用率 (single thread) キャッシュヒット率 0%
100 (デフォルト値) # dnsperf -s 192.168.11.21 -S 1 –c 1 -d query_same_list.txt -l 30	4,500QPS～ 7,500QPSでふらつく	100%
20 # dnsperf -s 192.168.11.21 -S 1 –c 1 -d query_same_list.txt -l 30 -q 20	4,500QPS– 4,900QPS	99 - 100%
10 # dnsperf -s 192.168.11.21 -S 1 –c 1 -d query_same_list.txt -l 30 -q 10	4,500 – 4,600 QPS	96 – 98%
5 # dnsperf -s 192.168.11.21 -S 1 –c 1 -d query_same_list.txt -l 30 -q 5	3,800QPS – 4,000QPS	88% - 92%

resperf

- resperfでテスト (60秒かけて 100,000QPS)
  - # resperf -s 192.168.11.21 –d query_random_list.txt
- resperf結果
  - [Status] Reached 65536 outstanding queries ← Transaction IDを全て消費したため終了
  - Statistics:
  - Queries sent: 96034
  - Queries completed: 96034
  - Queries lost: 0
  - Run time (s): 100.000001
  - Maximum throughput: 9582.000000 qps ← dnsperfの結果と大きく差がある
  - Lost at that point: 0.00%

- Maximum throughputとは?
  - テスト中に得られた最大 responses_per_sec
    - Maximum throughput が9,582 qpsのときは遅延が発生しているかもしれない

resperf

- どこらへんが限界値(QPS)か判断が難しい
  - 実際にDNSサーバが処理できる値に対して、resperfの負荷(60秒かけて10万QPS)が高すぎた。。

- unbound-controal stats結果 (QPS)
  - # while :; do unbound-control stats | grep 'total.num.queries'; sleep 1 ;done
  - total.num.queries=4301
  - total.num.queries=6358
  - total.num.queries=6621
  - total.num.queries=9400

- CPU 100%
  - PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
  - 760 unbound 20 0 135m 27m 2256 R 100 3.7 14:21.43 unbound

resperf

- resperfの結果ファイルresperf.gnuplot(テキストファイル)を確認
  - テスト終了後にresperfを実行したディレクトリに生成される
  - time: 0.5秒毎にプロット
  - target_qps: 目標qps
  - actual_qps :実際のqps
  - responses_per_sec: 戻ってきたレスポンス
  - failures_per_sec: failureレート (NXDOMAIN, NO ERROR を除くレスポンス)
  - avg_latency: 平均レイテンシ
- resperf.gnuplot 結果ファイル

```
# time target_qps actual_qps responses_per_sec failures_per_sec avg_latency
  0.250   416.67    416.00     416.00       0.00 0.000629
  0.750  1250.00   1250.00  1250.00     0.00 0.000519
  1.250  2083.33   2082.00  2082.00     0.00 0.000678
```

resperf

- actual_qps と responses_per_sec に差がでたところ
- avg_latency (　例えば、0.1以上, 1秒以上になったところ)

- # time target_qps actual_qps responses_per_sec failures_per_sec avg_latency
  - 3.250 5416.67 5418.00 5418.00 0.00 0.018137
  - 3.750 6250.00 6250.00 6250.00 0.00 0.102658 ←レイテンシ 0.1秒以上
  - 4.250 7083.33 7082.00 7082.00 0.00 0.290861
  - 4.750 7916.67 7918.00 7918.00 0.00 0.627598
  - 5.250 8750.00 8750.00 8750.00 0.00 1.109379 ← レイテンシ 1秒以上
  - 5.750 9583.33 9582.00 9582.00 0.00 1.746459 ← Maximum throughputの値
  - 6.250 10416.67 10418.00 9320.00 0.00 2.557626 ← actual_qps とresponses_per _secに差
  - 6.750 11250.00 11250.00 4110.00 0.00 3.058428

- 5,000 - 10,000QPSのどこか?

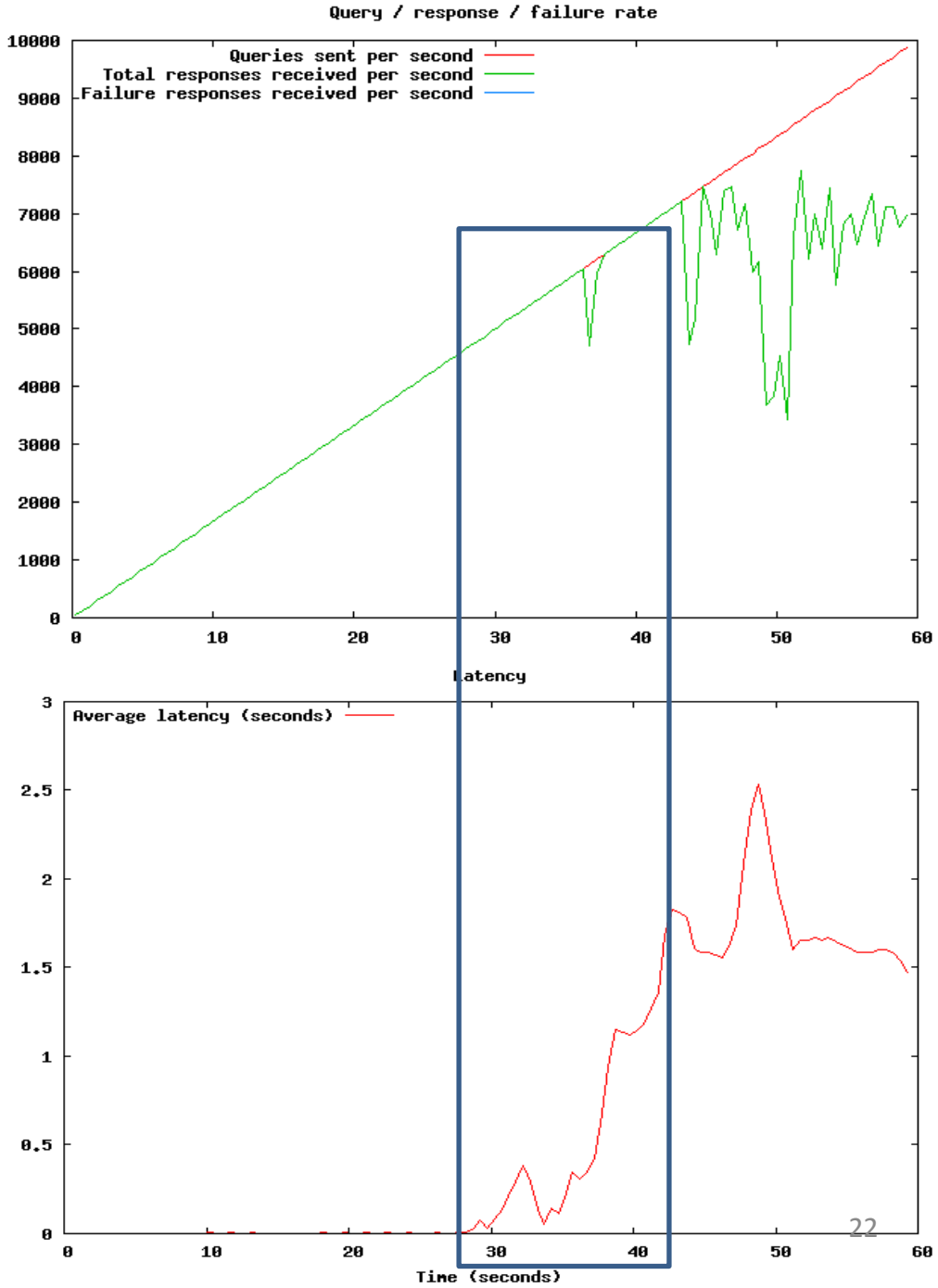
resperf

- 負荷をなだらかにしてテスト
  - -m 10000 -r 60
    - 60秒かけて、10,000QPS
- 結果ファイルを resperf-report でグラフ化
  - オプションは resperf も resperf-report 同じ
  - gnuplotをインストールする必要がある

```
# resperf-report -s 192.168.11.21 –d
  query_random_list.txt -r 60 -m 10000
Done, report is in 20131109-1901.html
```

resperf 結果グラフ

- 右上のグラフ: QPS
  - 赤: 送信クエリ
  - 緑: 受信
  - 青: Failure
- 右下のグラフ: レイテンシ
- レイテンシが上昇し始めたところのQPS
  - 5,000QPS – 5300QPS前後が低遅延で処理できる限界
- unbound上
  - QPS (増え方がなだらかになった)
    - total.num.queries=4709
    - total.num.queries=4900
    - total.num.queries=5316
  - CPU 90 – 96%



dnsperf, resperf

- dnsperf, respserf でもだいたい同じ結果になった
  - 無理やり5,000QPSあたりでまとめた(?)

- 挙動の違いを知っているとだいたい同じ結果を導けそう
  - dnsperf
    - -q, -c, -b の調整
  - resperf
    - ランプアップの調整
      - -r 60 –m 100000, -r 60 –m 10000, -r 60 –m 5000
    - -b の調整
    - 結果ファイル (resperf.gnuplot)を確認
    - actual_qps, reosponses_per_sec, avg_latency
    - maximum throughput の QPSだけで判断しない

参考:遅延エミュレート

- Linux Traffic Control
  - Ubuntu 12.04 LTS 64bit
- 100msec遅延追加
  - # tc qdisc add dev eth0 root netem delay 100ms
- 変更 (100msec → 10msec)
  - # tc qdisc change dev eth0 root netem delay 10ms
  - 変更するときは、最初に add している必要がある
- 削除
  - # tc qdisc delete dev eth0 root netem delay 10ms
- 設定確認
  - # tc qdisc show dev eth0

参考:dnsperf, resperf コンパイル

- Nominum dnsperf / resperf
  - dnsperf-src-2.0.0.0-1.tar.gz
    - http://nominum.com/support/measurement-tools/
    - または、nominu.com へブラウザでアクセス、dnsperf で検索

- コンパイル(Ubuntu12.04 64bit LTS)

  - # apt-get install libbind-dev build-essential dnsutils bind9 libpcap-dev tshark libxml2-dev libssl-dev libcap-dev gnuplot tshark

  - # ln -s /usr/lib/x86_64-linux-gnu/libgssapi_krb5.so.2.2 /usr/lib/x86_64-linux-gnu/libgssapi_krb5.so

  - # tar xzvf bind-9.9.4.tar.gz
  - # cp bind-9.9.4/lib/isc/include/isc/hmacsha.h /usr/include/isc

  - # tar xzvf dnsperf-src-2.0.0.0-1.tar.gz
  - # cd dnsperf-src-2.0.0.0-1/
  - # sh configure
  - # make
  - # make install

参考
dnsperf,resperfクエリファイルの書式

- queryperfと同じ書式
  - ドメイン　クエリタイプ

```
# cat query_list_sample.txt
www.foo  A
www.bar AAAA
foo.com MX
```

参考
Unbound

- unbound
  - unbound 1.4.16
  - # apt-get install unbound

- unbound.conf

  statistics-interval: 0

  statistics-cumulative: no

  num-threads: 1