新しいDNSサーバ、NSDの紹介

2004年8月30日

第124回 jus勉強会

伊藤 高一

kohi@iri.co.jp

NSDとは

- オランダのNLnet LabsがRIPE/NCCと共同で開発しているDNSサーバ。
- 2002年4月と書いてあるプレゼン資料に1.0.0-βが登場している。
- リリース日がはっきりしている範囲で一番古いのは2003年6月16日の1.0.3。
- 今回の資料は2.1.2(2004年7月30日リリース)について説明。
- h.root-servers.net、k.root-servers.netはNSDで運用されている模様。

NSDの特徴

- 速い。
  - NLnet Labsのプレゼン資料ではBIND 8の3倍速い、と主張。
- 動作をauthorityサービスに限定。
  - recursiveサービスはしない。
- シンプルな機能。
- BIND 4風の1ゾーン1行の設定ファイル。
- デフォルトでIPv6トランスポートが有効。

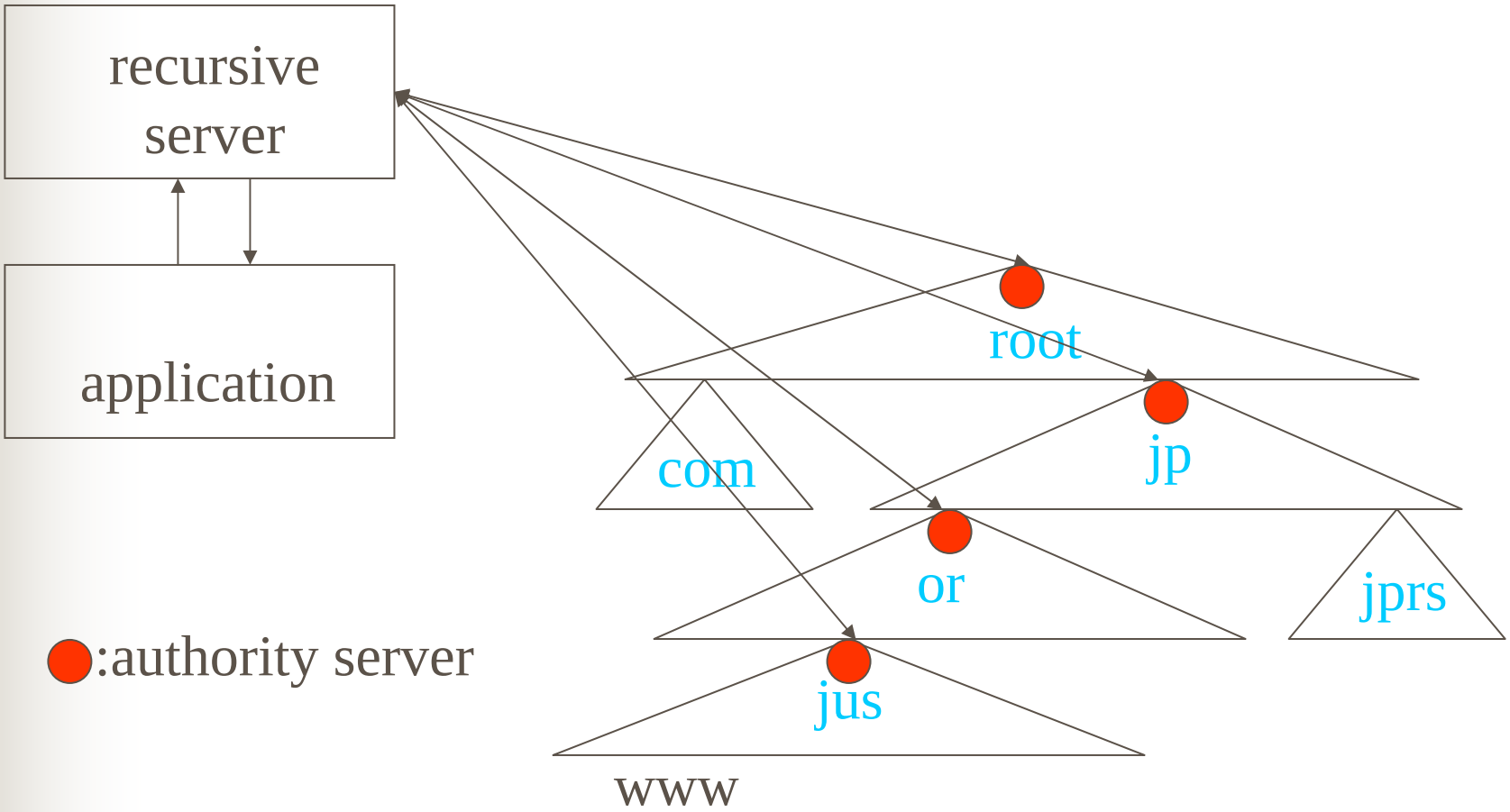
NSDの特徴(つづき)

- ゾーンデータは基本的にBINDと互換。
  - djbdnsは独自フォーマットでコンセプトも違う。
- 必ずrootの特権を放棄。
  - デフォルトではnsdというユーザにsetuid。
  - BINDは-uオプションをつけないとrootで動作。

authorityサービスとrecursiveサービス

- authorityサービス
  - world wideに対して、そのゾーンのネームサービスを提供する。
- recursiveサービス
  - 自サイトのクライアントに対して、world wideに関するネームサービスを提供する。

authorityサービスと recursiveサービス(つづき)

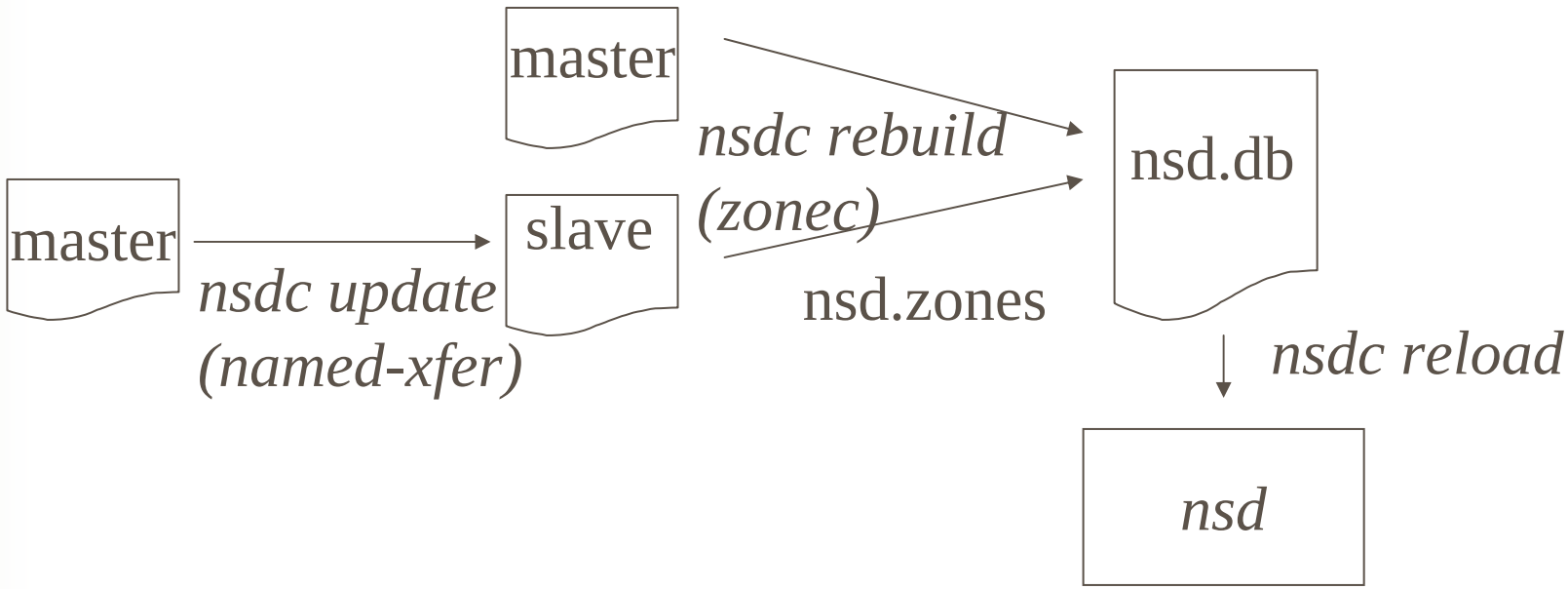


authorityサービスとrecursiveサービス(つづき)

- NSD
  - authorityサービスのみ。
- BIND
  - 1つのプロセスが両方のサービス。
  - recursion no;、allow-recursion{...}という設定あり。
- djbdns
  - authorityサービスはtinydns
  - recursiveサービスはdnscache

NSDのアーキテクチャ

- といってもサーバの内部構造ではなく、システムレベルの話ですが…。



nsdc

- (r)ndcと同じような制御コマンド
- start/stop/reload/rebuild/restart/running/update/notify
- reload
  - nsd.dbを読み直す。
- rebuild
  - テキストのゾーンデータ(群)からnsd.dbを生成。

nsdc(つづき)

- update
  - 内部でnamed-xferを起動して、slaveをしているゾーンについてmasterからゾーン転送。
  - 更新があればrebuld、reloadも自動的に実行。
- notify
  - notify(RFC1996)を送出する。
    - 本来はmasterが更新されたことをslaveに通知。
    - nsdcは更新の有無と無関係に送出する。

slave

- slaveをしているゾーンのデータの保守はnsdc updateで行う。
  - 例えばcronで起動。
  - SOAのパラメータは使われない。
    - masterも自分で管理しているときはよいが、他サイトのslaveをしているときなどは問題かも。
  - named-xferはnsdではなくnsdcの子。
    - chroot()環境にnamed-xferは不要。

インストールから起動まで

- ソースを入手/configure/make/make install
- FreeBSDだと1.2.4のportsあり
- masterになるゾーンのゾーンデータを用意
- nsd.zonesを作成
- 必要に応じてnsdc.confを作成
- nsdc update(slaveになるときだけ)
- nsdc rebuild
- 起動

ゾーンデータ

- ゾーンデータは
  - RFC1035フォーマット
    - BINDと基本的に互換
  - $TTL
    - RFC2308拡張
  - M,H,D,W
    - BIND 8以降の独自拡張
  - $GENERATEはサポートしていない。
    - これもBIND 8以降の独自拡張

TTLの扱い

- $TTLディレクティブがないと
  - BIND
    - SOAのminimumの値をdefault TTLに使う。
    - BIND 8.2以前とのbackward compatibility。
  - NSD
    - minimumとは無関係に1時間にする。
    - RFC1033のお勧めは1日
    - RFC1912のお勧めは1日〜5日

nsd.zones

- 1ゾーン1行のテキスト
  - `zone` *ゾーン名* *ファイル名*
  - `zone` *ゾーン名* *ファイル名* `masters` *IPアドレス…*
  - `zone` *ゾーン名* *ファイル名* `notify` *IPアドレス…*

nsd.zones(つづき)

- mastersパラメータ
  - nsdcはそのゾーンをslaveと見なす。
  - nsdc updateを実行すると*IP アドレス*からゾーン転送し、*ファイル名*に書き出す。
  - *IP アドレス*はnamed-xferにそのまま渡される。
    - BIND 8.3以前のnamed-xferだとv6アドレスは指定できない。

nsd.zones(つづき)

- notifyパラメータ
  - nsdc reload、nsdc notifyを実行すると、*IPアドレス*に向けてnotifyを送出する。
- nsd(サーバ)はnsd.zonesを見ない。
  - nsdが見るのはzonecで生成したnsd.db。
  - nsdはそのゾーンがmasterかslaveかは気にしていない。
  - nsdはゾーン転送もnotifyの送出もしない。

ゾーン転送(入り方向)

- nsdは行わない。
- nsdc update
  - 能動的には行わないので、cronなどで起動。
- リクエストにtsig署名できる。
  - *${NSDKEYDIR}/IP-addr-of-master*.tsiginfoというファイルに鍵を設定。
  - 鍵の指定はゾーン毎ではなくmasterサーバ毎。

ゾーン転送(出方向)

- ゾーン転送のアクセス制限はtcp_wrappersで実現。

```
axfr: all: deny
axfr-jus.or.jp.: 192.168.0.1: allow
```

- tsig認証はできなさそう。

nsdc.conf

- nsdcの設定ファイル
- nsdc updateで起動するnamed-xferのパス
- nsdc rebuildで生成するnsd.dbのパス
- nsdc startでnsdに与える引数

など

- configureで指定したパラメータで用が足りればnsdc.confは不要。

起動

- nsd [*options*...]
- **主なオプション**
  - -a *IP アドレス*
    - **特定のアドレスだけに**bind()
  - -N *子プロセス数*
  - -s *sec*
    - **定期的に**BIND 8**互換の統計情報をログに出力**
  - -t *chroot*

chroot()、setuid()

- chrootするときも各種パスはchroot()前のパスを指定する。
  - nsd -t /sandbox -f /sandbox/var/db/nsd.db
- BINDは、chroot()後のパスを指定。
  - named -t /sandbox -c /etc/named.conf
- NSD
  - pid書き出し;chown()->chroot()->setuid()
- BIND 9
  - chroot()->setuid()->pid書き出し
- BIND 8
  - chroot()->pid書き出し;chown()->setuid()

データ更新

- masterになっているゾーン
  - ゾーンデータ(テキスト)を編集
  - nsdc rebuild
  - nsdc reload
- slaveになっているゾーン
  - cronなどでnsdc update

ベンチマークの一例

- 被計測ホスト
  - Pentium 166MHz
  - FreeBSD 5.2.1
- 計測ホスト
  - Celeron 1.2GHz
  - FreeBSD 4.10
- スループットとレイテンシを計測

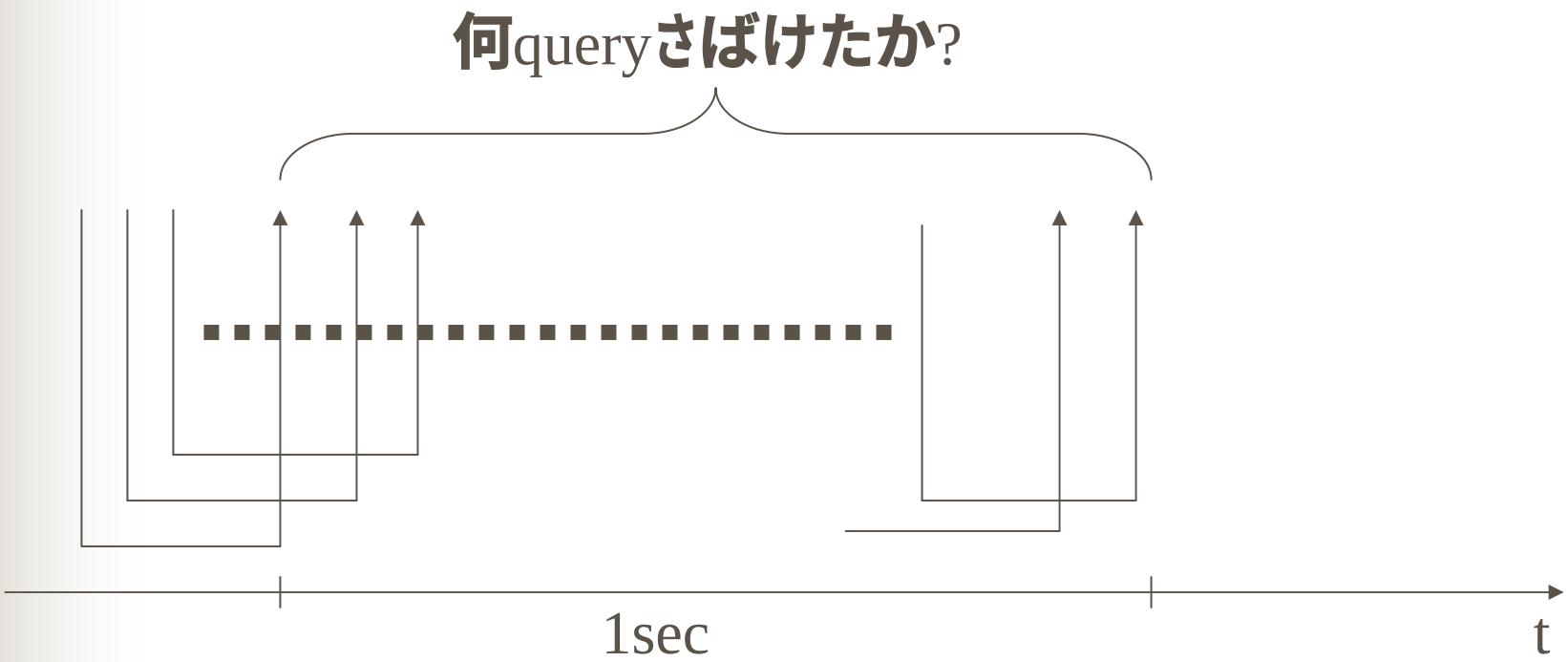
設定

- **設定したゾーン**
  - localhost
    - **ごく普通**
  - example.jp
    - `test-0-0 A 192.168.0.0`

      :
    - `test-255-255 A 192.168.255.255`
- BIND **は**
  - recursion no;
  - fetch-glue no;

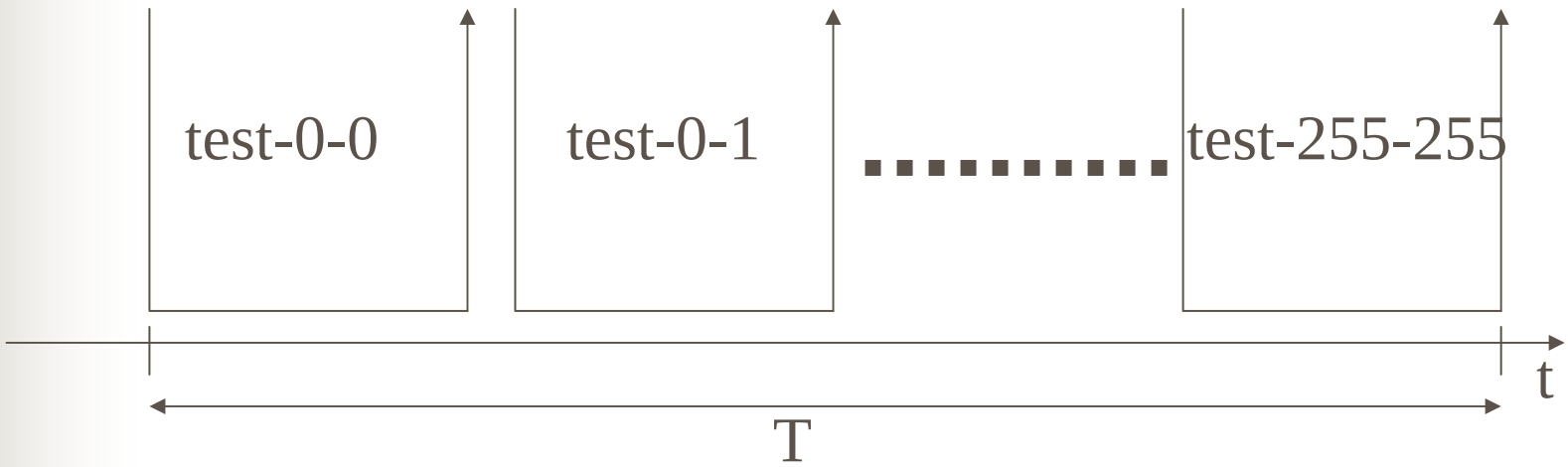
スループット

- BIND9/contrib/queryperfで計測



レイテンシ

- 自作プログラムで計測
  - gethostbyname()のループ
  - recursive desired = off



結果

実装	スループット (qps)	レイテンシ (sec)
NSD-2.1.2	1927.5	53
BIND 8.3.7-REL	1583.8	65
BIND 9.2.3	938.9	89

- 参考(N+Iでのかとうさんのプレゼンから)
  - m.root-servers.net: 5〜8kqps
  - e.dns.jp: 0.7〜1kqps

ps:ゾーンデータあり

```
# UID   PID  PPID CPU PRI NI   VSZ  RSS MWCHAN STAT  TT       TIME
COMMAND
% ps alxw | grep nsd | egrep -v '(grep|syslogd)'
   53 70842     1  53   8  0  9036 8596 wait   Is    ??    0:09.77
/u1/nsd-2.1.2/sbin/nsd -f /u1/nsd/etc/nsd.db
   53 70854 70842  52 102  0  9036 8596 select I     ??    0:00.00
/u1/nsd-2.1.2/sbin/nsd -f /u1/nsd/etc/nsd.db


% ps alxwp `cat /var/run/named.pid`
  UID   PID  PPID CPU PRI NI   VSZ  RSS MWCHAN STAT  TT       TIME
COMMAND
    0 74378     1   0  96  0  7728 7072 select Is    ??    0:00.01
/usr/sbin/named -c /u1/bind/etc/named.conf


% ps alxwp `cat /var/run/named.pid`
  UID   PID  PPID CPU PRI NI   VSZ  RSS MWCHAN STAT  TT       TIME
COMMAND
    0 74397     1   3  96  0  9776 9256 select Ss    ??    0:18.68
/usr/local/bind-9.2.3/sbin/named -c /u1/bind/etc/named.conf
```

ps:ゾーンデータなし

```
# UID   PID  PPID CPU PRI NI   VSZ  RSS MWCHAN STAT  TT       TIME
COMMAND
% ps alxww | grep nsd | egrep -v '(grep|syslog)'
   53 77292     1   0   8  0  1320  872 wait   Is    ??    0:00.01
/u1/nsd-2.1.2/sbin/nsd -t /u1/nsd
   53 77293 77292   0  96  0  1320  872 select I     ??    0:00.00
/u1/nsd-2.1.2/sbin/nsd -t /u1/nsd


% ps alxwwp `cat /var/run/named.pid`
  UID   PID  PPID CPU PRI NI   VSZ  RSS MWCHAN STAT  TT       TIME
COMMAND
    0 77271     1   0  96  0  2572 1780 select Ss    ??    0:00.01
/usr/sbin/named -c /u1/bind/etc/named.conf


% ps alxwwp `cat /var/run/named.pid`
  UID   PID  PPID CPU PRI NI   VSZ  RSS MWCHAN STAT  TT       TIME
COMMAND
    0 77277     1   0  96  0  2652 2020 select Ss    ??    0:00.08
/usr/local/bind-9.2.3/sbin/named -c /u1/bind/etc/named.conf
```

ps(つづき)

- ゾーンデータを読まないと小さい。
- ゾーンデータを読むと大きくなる。
  - データ構造に起因? 速さと引き換え?
- 親は残る。
  - 多分、制御を受け持つ。
- -Nで指定した個数(デフォルトは1)fork()する。
  - 多分、queryをさばいているのはこいつら。

BINDからの移行

- ゾーンデータは$GENERATE以外、そのまま。
  - $GENERATEを使っているなら外部で展開する。
- NSDにない機能を使っていると無理。
  - dynamic update、view、queryのアクセス制限…
- 1プロセスでauthorityサーバ、recursiveサーバを兼ねている場合
  - ifconfig fxp0 192.168.0.1 netmask 255.255.255.224
  - ifconfig fxp0 192.168.0.2 netmask 255.255.255.255 alias
  - nsd -a 192.168.0.1
  - named.conf:listen-on{192.168.0.2;localhost;};

しばらく使って/調べてみた感想

- 手許の環境では「3倍」は出なかったが、確かに速い。
- 設定は簡単。
- $GENERATE以外、ゾーンデータはBIND互換なのは、おいしい。
- 機能を絞り込んでいる分
  - 適用箇所が限定される。
  - 使いこなしは難しいかも。

向き、不向き

- アクセスの多いサイトのauthorityサーバにはよさそう。
- xSPで使うなら…
  - 自社設備に関するゾーンのauthorityサーバにはよさそう。
  - poolアドレスとかRFC2317(/24に満たないアドレス空間の逆索き)の親とかは$GENERATEがネック。
  - ホスティングにもよさそう。

向き、不向き(つづき)

- slaveとしての動作にSOAのパラメータが反映されないので、顧客のslaveをするのには向いていない。
- 小規模サイトのall-in-oneネームサーバには
  - recursiveサーバとの分離
  - split DNSやアクセス制限などの凝った設定

  がネック。

参考URL

- http://www.nlnetlabs.nl/nsd/
  - **本家のページ**
- http://www.nic.ad.jp/ja/materials/iw/2003/main/dns/2-1-morishita.pdf
  - BIND 9、djbdns(tinydns)、NSD**を比較している。**