NSD

山口崇徳@IIJ

DNS Summer Day 2016

自己紹介

- IIJというところにいます
  - サポート屋さんにジョブチェンジしました
  - 運用のお仕事はほぼ引退

- 最初のDNSのお仕事は BIND4 → BIND8 の移行
  - 前世紀末
- ほかに DNS を担当する人がいなくて押しつけられただけだったのに、気がつけばはや幾年…

- お仕事では NSD は(まだ)使ってません
  - プライベートでは 2.0 のころから(もう10年以上)

NSDとは?

- NLNetLabs と RIPE NCC 共同開発による権威 DNS サーバ
  - http://www.nlnetlabs.nl/projects/nsd/
- [HKL].root-servers.net などで稼動
  - ルートサーバでも稼動実績がある、というより、ルートサーバで使うことがそもそもの開発の目的
- BSD ライセンス
- 現在の最新版は 4.1.10
  - 3.x は今年5月に EoL (その後 3.2.22 が出たけど…)

NSD の特徴

- よく使われる必要な機能だけに絞った実装
  - 実装された機能の「数」は少ないが、実装済み機能の「質」についてはほぼ十分
- 機能が少ない → コードが複雑化しない
  - 少ない脆弱性
  - 高いパフォーマンス

- NSD で機能が足りない場合は KnotDNS がいいかも
  - けっこう何でもできるらしい(伝聞)
  - パフォーマンスも NSD よりいいらしい(伝聞)

BINDとの違い

- キャッシュサーバ機能なし
  - 同じ NLNetLabs 製の Unbound が担当
- view なし
- dynamic update なし
- DNSSEC 鍵管理/署名機能なし
  - DNSSEC 自体は対応している
- クエリログ取得不可

- BIND にできて NSD にできないことはたくさんあるが、NSD にできて BIND にできないことはほとんどない

Unbound と似てるところ違うところ

- 開発はどちらも NLNetLabs
- NSD: 権威専用 / Unbound: キャッシュ専用
- N: マルチプロセス / U: マルチスレッド
- N: 必要最小限の機能 / U: わりと機能てんこ盛り
- 設定ファイルの形式はほぼ同じ
  - ただし、同じ意味の設定でパラメータ名が異なることがある
    - N: ip-address, round-robin, reuseport, statistics
    - U: interface, rrset-roundrobin, so-reuseport, statistics-interval
- 遠隔制御({nsd,unbound}-control) はそっくり
  - ただし、Unbound は設定ファイルの更新なしで設定をかなり自由に変更できるが、NSD はゾーン追加/削除ぐらい

セキュリティ

- これまで脆弱性がなかったわけではないが、見つかる頻度は小さい
  - CVE-2009-1755 DoS/コード実行
  - CVE-2012-2978 DoS
  - CVE-2012-2979 DoS (非標準 configure オプション有効時のみ)
  - もちろんいずれも修正済み

ベンチマーク(参考)

- CPU 2GHz 4core の仮想環境
- A レコード1000個入りゾーンを100個ロード
- dnsperf で100万クエリ送出
  - クエリのうち、ゾーンに存在する名前と存在しない名前の割合 1:1

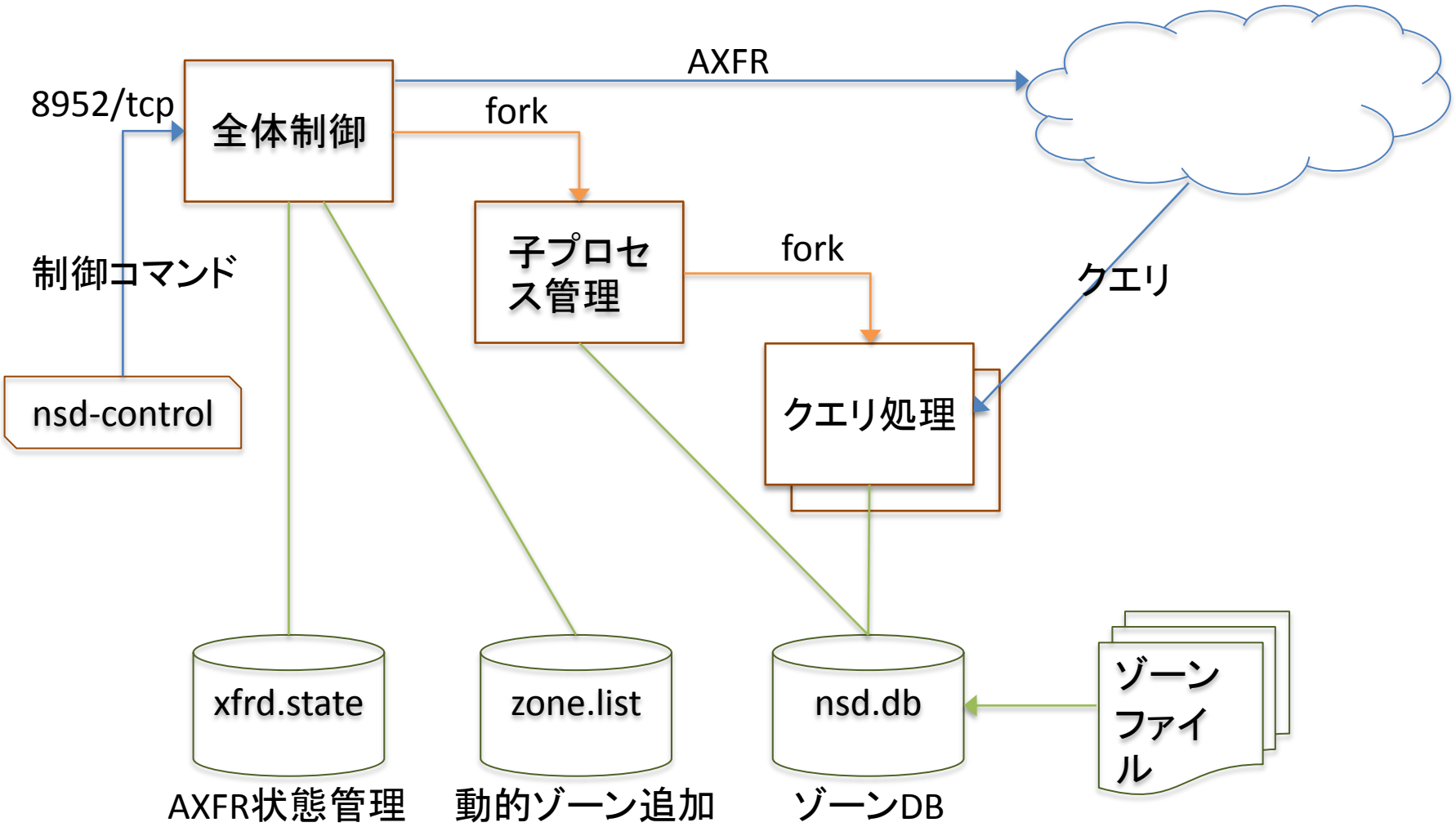
	server-count	**qps**	**備考**
BIND 9.10.3-P4	自動設定	29889	CPU 100%
NSD 4.1.9	1	88527	デフォルト; CPU 100%
	2	86906	CPU 使いきってません…
	4	81136	同上

- server-count: 1 (デフォ)で BIND の3倍程度
- server-count: 2以上では負荷をかける側の性能が足りなくて、サーバの CPU が暇してました…
  - 理屈では CPU のコア数と server-count を一致させるともっとも性能がよいはずだが確認できず

パフォーマンス

- NSD はたしかに BIND より性能が高い
  - KnotDNS はもっと性能が高いらしい
- が、今どきのマシンなら、多くの場合 BIND でも性能は十分
  - DDoS を食らったとしても、昨今はネットワークの方が先に限界が来ることが多い
- パフォーマンスを理由に BIND から乗り換えるのは、今よほど困ってるわけでもないかぎりオススメしない
  - もっと他の理由を探すべし

NSD のアーキテクチャ



nsd.db

- ゾーンファイルをそのまま解釈するのではなく、いったん「コンパイル」してバイナリ形式に変換してからロードする
  - ゾーンの読み込みの高速化
  - 3.x までは手動コンパイルが必要だったが、4.x では不要に
- 4.1.0 から nsd.db を使わない設定も可能になった

```
database: ""      # 空文字列
```

  - slave としての利用が主の場合、master からのゾーン取得に時間がかかるので、コンパイルによる時間短縮のメリットが薄い
  - 手元のテスト環境ではメモリ消費量約4割減
  - nsd.db の使用有無にかかわらず、ゾーン情報はすべてメモリ上にロードされるため、応答速度は変わらない

チューニング(1)

- パフォーマンス
  - server-count
    - クエリ処理用の子プロセスの数(default 1)
    - CPU コア数と同じ値に設定するとよい
    - 起動時の引数でも指定できる(-N)
  - reuseport
    - set_sockopt(2) で SO_REUSEPORT を有効にする(default no)
    - server-count が2以上でないと yes にしても意味なし
    - 最近の Linux 専用(*BSD ではうまく動かないらしい)
      - Unbound の so-reuseport は FreeBSD や OSX でも may also work だそうな

チューニング(2)

- ネットワーク
  - EDNS まわり
    - ipv[46]-edns-size: EDNS の最大サイズ
  - TCP まわり
    - tcp-count: TCP の最大同時接続数
    - tcp-timeout: TCP タイムアウト(ゾーン転送含む)
    - tcp-mss, outgoing-tcp-mss
    - など
  - 規模に応じて tcp-count を調整
    - ほかはあまり気にしなくてよいかと

IP エイリアスの注意点

- ひとつのインターフェイスに複数の IP アドレスを割り当て
  - ロードバランサによる冗長化
  - anycast 構成
  - /32 ホストルーティング
- デフォルトでは 0.0.0.0 にバインドする(ip-address: 0.0.0.0)
- が、IP エイリアス使用時はかならずアドレスをひとつずつ列挙するように設定を修正すること
  - 0.0.0.0 のままだと、仮想 IP 宛のクエリに対して、実 IP をソースアドレスとする応答を返してしまう

ログ

- BIND ほど凝ったログは取得できない
  - 出力先の設定(syslog かファイルか)
    - syslog の facility は LOG_DAEMON 固定で変更不可
  - ログの冗長度(verbosity)
  - …ぐらい
- クエリログも取れない
  - 付属ドキュメントにて実装しないと明言
  - 必要ならパケットキャプチャを

nsd-control (1)

- BIND における rndc のようなもの

```
remote-control:
    control-enable: yes
```

  - 設定の読み直し(reconfig)
  - ゾーンの読み直し(reload)
  - ゾーン転送まわり(notify/transfer/force_transfer)
  - 動的ゾーン追加(addzone(s)/delzone(s))
  - ステータス/統計(status/zonestatus/stats/stats_noreset)
  - など
- 8952/tcp で TLS 接続
  - nsd-control-setup でサーバ証明書/クライアント証明書を生成
  - オレオレ証明書だけど、この用途では安全性に問題はない

nsd-control (2)

- nsd-control zonestatus が便利
  - 保持しているゾーンの列挙
  - slave なら master との同期状況も見られる
  - 数少ない「NSD にできて BIND にできないこと」のひとつ

```
# nsd-control zonestatus
zone:   example.com
        state: master
zone:   example.org
        state: ok
        served-serial: "2016030801 since 2016-05-31T10:51:01"
        commit-serial: "2016030801 since 2016-05-31T10:51:01"
```

- rndc start はできないけど nsd-control start はできます

パターン

- テンプレートといった方が通じやすいかも?
- 似たような設定のゾーンがいくつもあるときに便利

```
pattern:
    name: "master"
    zonefile: "master/%s.zone"          # %s にゾーン名が入る
    notify: 192.0.2.1
    provide-xfr: 192.0.2.1 NOKEY
pattern:
    name: "dnssec"
    include-pattern: "master"                   # パターンの多重適用も可
    zonefile: "master/%s.signed"
zone:
    name: "example.com"
    include-pattern: "master"
zone:
    name: "example.jp"
    include-pattern: "dnssec"
```

- "nsd-checkconf -v nsd.conf" でパターンの展開結果を確認可
  - が、zonefile 中の %s は展開してくれない…

動的ゾーン追加

- nsd.conf を修正せずにゾーンの追加・削除が可能
  - 新規追加するゾーンが適用されるパターンを事前に用意しておく必要あり
    - BIND のように任意の設定を投入できるわけではない
  - example.com ゾーンを追加し、hoge パターンを適用する

```
# nsd-control addzone example.com hoge
```

  - example.jp ゾーンを削除

```
# nsd-control delzone example.jp
```

- addzone/delzone したゾーンの情報は nsd.conf ではなく、zone.list に格納される
  - プロセス終了後も内容は保持される
  - zone.list を読むのは起動時だけ
    - 稼働中に手で修正して nsd-control reconfig しても反映されない

ゾーンファイルの記述(1)

- ほぼ RFC1035 準拠
  - BIND と同じゾーンファイルがそのまま使える、ということ
  - RFC2308 拡張の $TTL は使える
  - BIND 独自拡張の $GENERATE は使えない
- アンダースコア(_) の含まれるホスト名などをゾーンファイルに書くと BIND ではエラーになるが、NSD では素通りする
  - DNS の仕様上はアンスコは許容されている
  - DNS とは別のところ(RFC1123)でアンスコ入りホスト名が禁止
- その他、ゾーンの記述チェックは BIND より緩い
  - MX レコードが指してるホスト名が CNAME だった場合に BIND は警告を出すけど NSD は出さないとか

ゾーンファイルの記述(2)

- RRSet 内の各 RR で TTL の異なるものを記述

```
www  100  IN  A  192.0.2.1
www  200  IN  A  192.0.2.2
```

  - RFC2181 section 5.2 曰く、「やっちゃダメ。クライアントはこういう応答を受けとったらエラーにすべき」
    - ほんとにエラーにする実装の存在は未確認
  - BIND はゾーンファイルで先に書かれているもの(上の例では100)に値を揃えて応答する
  - NSD は矯正してくれず、ゾーンファイルに書いてあるとおりの TTL で応答する
- ゾーンファイルを書く人間が注意する必要あり

named-checkzone

- named は捨てても named-checkzone は捨てるな
  - $GENERATE の展開、ゾーンの記述チェック、不揃い TTLの修正など、いずれも named-checkzone におまかせできる

```
named-checkzone -D example.com example.com.zone
```

  - -D: ゾーンファイルを正規化して出力
- named-compilezone でも可

```
named-compilezone -o file example.com example.com.zone
```

  - コマンドの実体としては checkzone と同じもの
  - named-compilezone という名前で起動すると、-D が暗黙に指定されるかわりに、出力ファイル名(-o)を省略できなくなる
  - その他いくつかのチェック項目は compilezone の方が厳しい
- 詳細は man named-checkzone

master - slave の連携(1)

- ゾーン転送まわりは全体的にあまりイケてない
  - とくに slave 側

- 差分転送(IXFR)
  - master 側は非対応
  - slave 側は対応
  - NSD を master に使うと転送サイズが増える
- 万が一 master でおかしな SOA refresh/retry を指定されてしまった場合、slave 側でそれを矯正する手段がない
  - 困ったのでパッチ書いて取り込んでもらいました
  - 次のリリース(4.1.11?)から使えるようになるはず
  - {max,min}-{refresh,retry}-time

master - slave の連携(2)

- BIND
  - slave はまず master の SOA serial が大きいかどうかを確認してからゾーン転送をはじめる
  - master が複数ある場合すべて確認してもっともシリアルの大きいものにしたがう
- NSD
  - SOA を確認せずいきなり AXFR(IXFR) を送りつけ、応答に含まれる SOA serial が更新されていなかったら転送途中で切断する
  - master が複数ある場合、最初に見つけた自分よりシリアルの大きな master にしたがう
    - master のそれぞれでシリアルが一致していない場合、slave のシリアルがもっとも大きなものにならない可能性がある
    - ちなみに KnotDNS もダメらしい……

master - slave の連携(3)

```
// BIND
zone "example.com" {
    type master;
    file "example.com.zone";
};
```

```
# NSD
zone:
    name: "example.com"
    zonefile: "example.com.zone"
```

- どちらも同じような設定に見えるが……
- BIND はデフォルト(notify yes)では、とくに指定しなくてもゾーンファイル中の NS レコードのホストに NOTIFY を送る
- NSD はアドレスを明示的に指定しないかぎり NOTIFY を送ることはない
  - BIND の notify explicit 相当の動作
  - notify yes 相当の動作にする設定はない

ラウンドロビン

- ちょっと前のバージョンまで、「実装するつもりないよ」とドキュメントで明言されていた
- が、4.1.0 から対応
- デフォルトではこれまでどおりラウンドロビンしないので、必要なら明示的に設定する

```
round-robin: yes
```

- Unbound もデフォルトではラウンドロビンしない
  - デフォルトの Unbound からデフォルトの NSD に問い合わせがあると、応答順が常に固定になってアクセスが偏ることになる

DNSSEC

- ちゃんと対応してます
- が、署名鍵を作成したり、ゾーンファイルに署名したりする機能/ツールはない
- NSD とは別のところで鍵を作って署名したゾーンを NSD に読ませればよい
- 例
  - OpenDNSSEC
  - BIND の dnssec-keygen、dnssec-signzone コマンド
  - ldns の ldns-keygen、ldns-signzone コマンド
  - inline-signing 機能を有効にした BIND を master にして、NSD をその slave にする
  - など

RRL

- Response Rate Limting
  - 権威サーバにおける DNS amp 対策のひとつ
  - キャッシュサーバがキャッシュしてる(はずの)情報を繰り返し問い合わせてくるような場合に応答を制限する仕組み
- パフォーマンスがよい = DNS amp の踏み台としても優秀

- 詳細は略
  - http://dnsops.jp/event/20130529/dnssec2013springforum-yamaguchi-1.pdf あたりを参照
    - 3.2 のころの情報なのでちょと古いですが…
    - 4.1 では slip の割合その他設定できる項目が増えている

まとめ

- シンプル
  - 機能少ない
    - とくに slave
    - 特殊なことをやらないのであれば、たいてい必要な機能は揃っている
  - 脆弱性少ない
  - 速い
- 足りない機能を補うツールがあると便利
  - named-checkzone や DNSSEC まわりなど