Validns – www.validns.net

a high performance DNS /
DNSSEC zone file validator

2012 - Chrischurch, New Zealand

Anton Berezin
Out of Band Consulting

Phil Regnauld
Network Startup Resource Center

Who are we

Anton Berezin, freelance developer

- Perl hacker, C when arm is twisted the right way
- Author of TIPP IP Planner, BSDPAN

Phil Regnauld, trainer and systems architect

- Self qualified hardcore generalist
  - DNS, sysadm, network management

Background

- A discussion thread on the OARC dns-operations mailing list in March 2011
- Recent incidents (read : bugs) in DNSSEC signing software
- Call for a tool that could test *large* signed RFC1035 (« BIND ») format DNS zones
- Existing zone validation tools or services :
  - Too slow or never terminate
  - Incomplete support for DNSSEC

https://lists.dns-oarc.net/pipermail/dns-operations/2011-March/006919.html

Requirements

- Open Source
- Run on *N*X
- Highly desirable that it should complete the task without crashing, eating all memory
- And preferably before heat death of universe

Checking a zone

- Syntactical validity

```
baa.nz.     MX  10  1.2.3.4
mutant.ip.  A       192.168.1.2.3
```

- Names

```
bad#!&beer.   TXT “budweiser”
```

Checking a zone (2)

- Semantics

```
www.zone.       CNAME   server.zone.
// missing server.zone.
```

- Policy

```
zone.       NS      ns1.zone.
// only one NS
```

DNSSEC

- DNSSEC complicates things
  - Increased complexity
  - Harder to parse zone visually
  - NSEC/NSEC3 introduces dependencies between records (chains)

DNSSEC (2)

- New requirements
  - At least 1 set of matching keys (KSK, ZSK) for each of the signatures covering Rrs in this zone ?
  - Are the signatures valid timewise ? (inception, expiration)
  - Is the NSEC chain valid ?

No tool available to test this...

validns

- Requirement specification done a few days after initial mail on dns-operations
- First functioning version a week later
- Written in C for performance
- Doesn't reuse existing DNS parsing libraries
  - Possibly some new bugs...
  - But newer code also means greater diversity and robustness in the « cross examination » of the zone data

Scope

- Scope of the testing : the zone, nothing but the zone
  - Not testing whether DS is published in parent
  - Not testing visibility of keys on public Nses
  - Not testing if a key is ready to be retired
  - Not testing if the contents of the website pointed to by the domain infringe on someone's intellectual property

Catching errors

- Let's take a look at some examples...
  - Chained CNAMEs
  - Dangling CNAMEs
  - Invalid syntax
  - Missing dots [?]
  - Invalid hash, invalid date

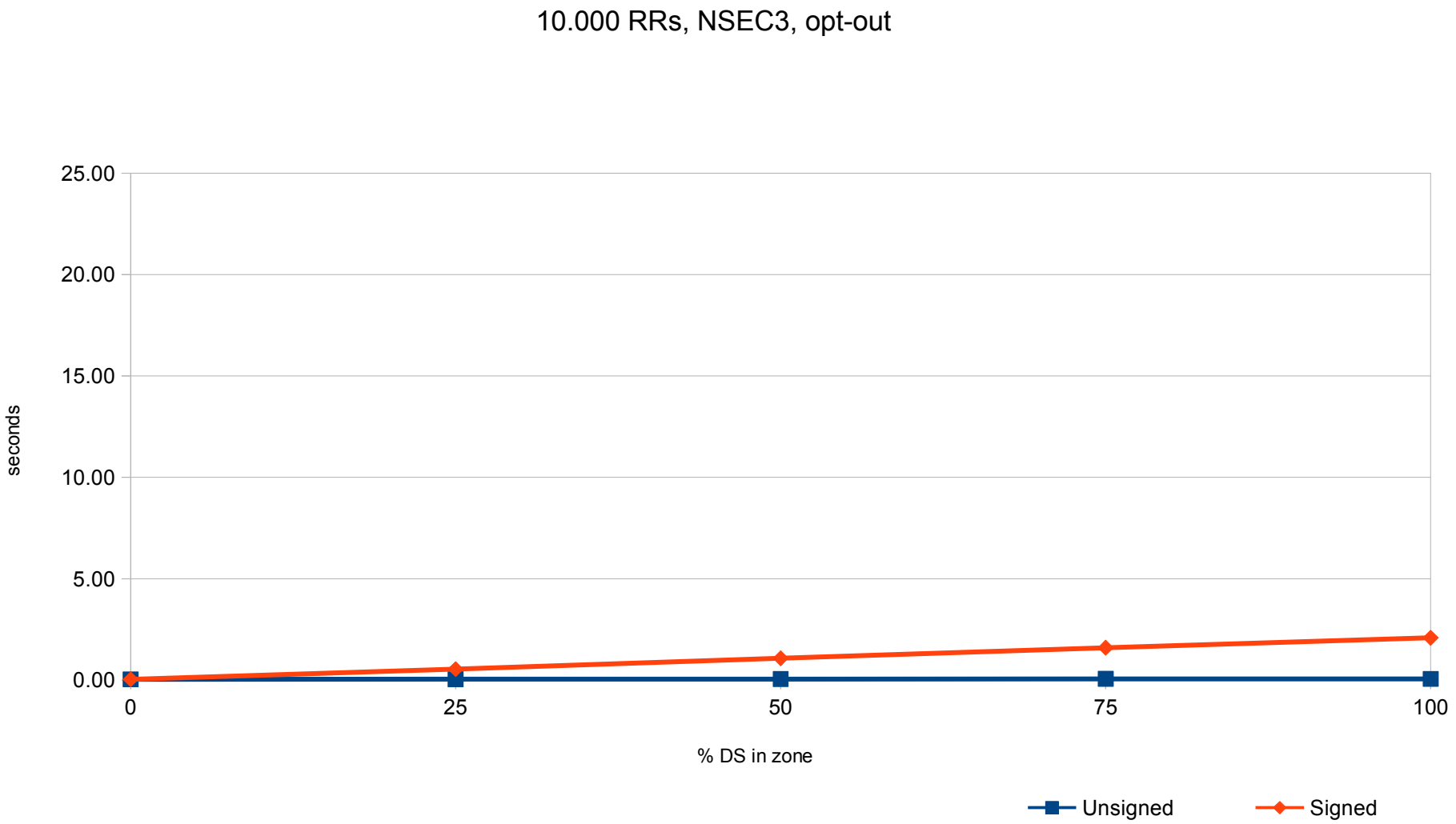
Invocation

```
Usage:
   validns -h
   validns [options] zone-file
Usage parameters:
      -h          produce usage text and quit
      -f          quit on first validation error
      -p name     perform policy check <name>
                  single-ns
                  cname-other-data
                  dname
                  nsec3param-not-apex
                  mx-alias
                  ns-alias
                  rp-txt-exists
                  all
      -q          quiet - do not produce any output
      -s          print validation summary/stats
      -v          be extra verbose
      -I path     use this path for $INCLUDE files
      -z origin   use this origin as initial $ORIGIN
      -t epoch-time use this time instead of "now"
```

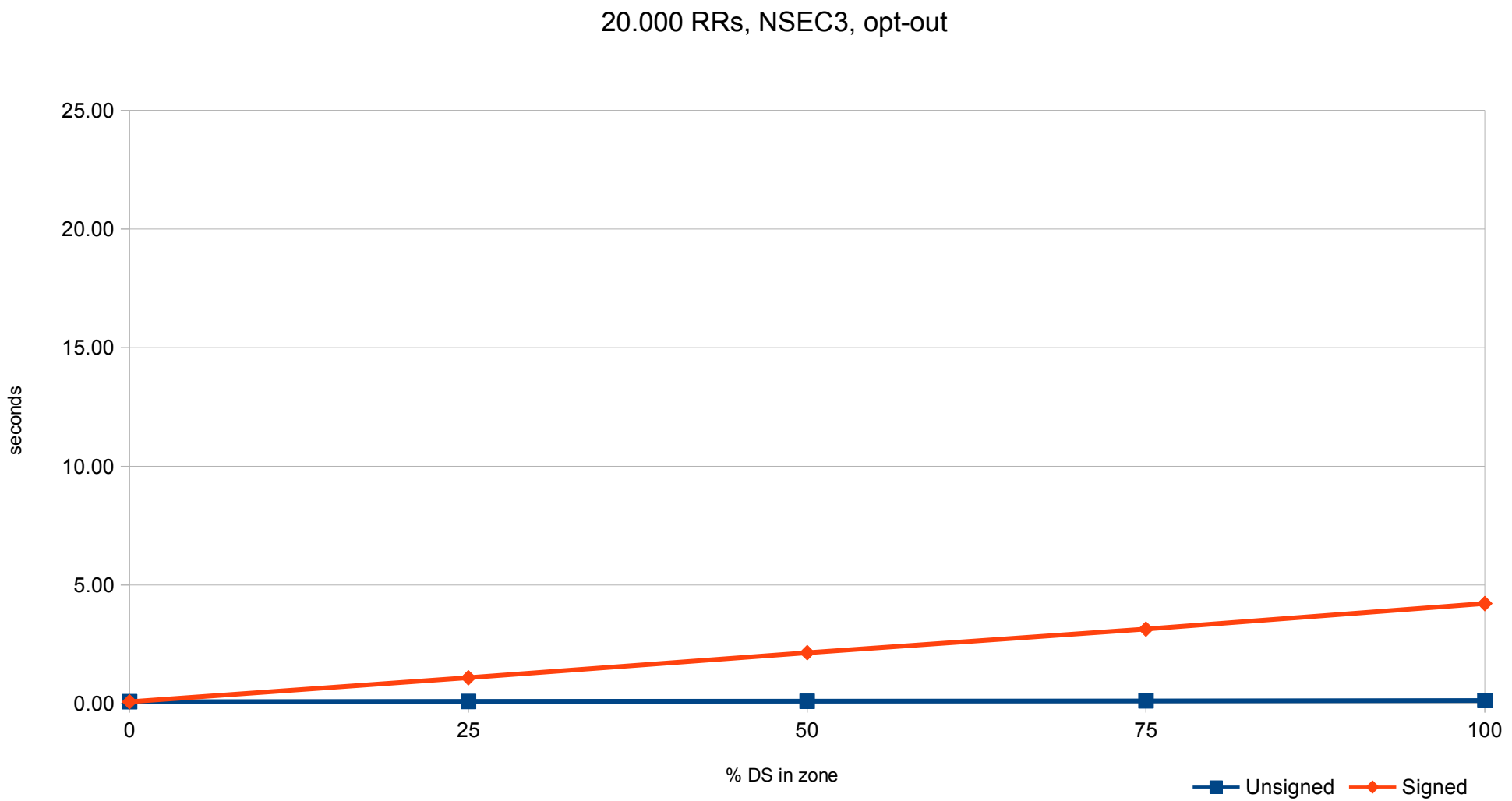
Performance

- A few random figures
  - 4.3 M RRs, 40 signatures in 8.5 seconds
  - 4.2 M RRs, 515 kSigs + 340 kNSEC3 in 48.5 seconds
  - 200 k RRs, 100 kSigs in 5.7 seconds

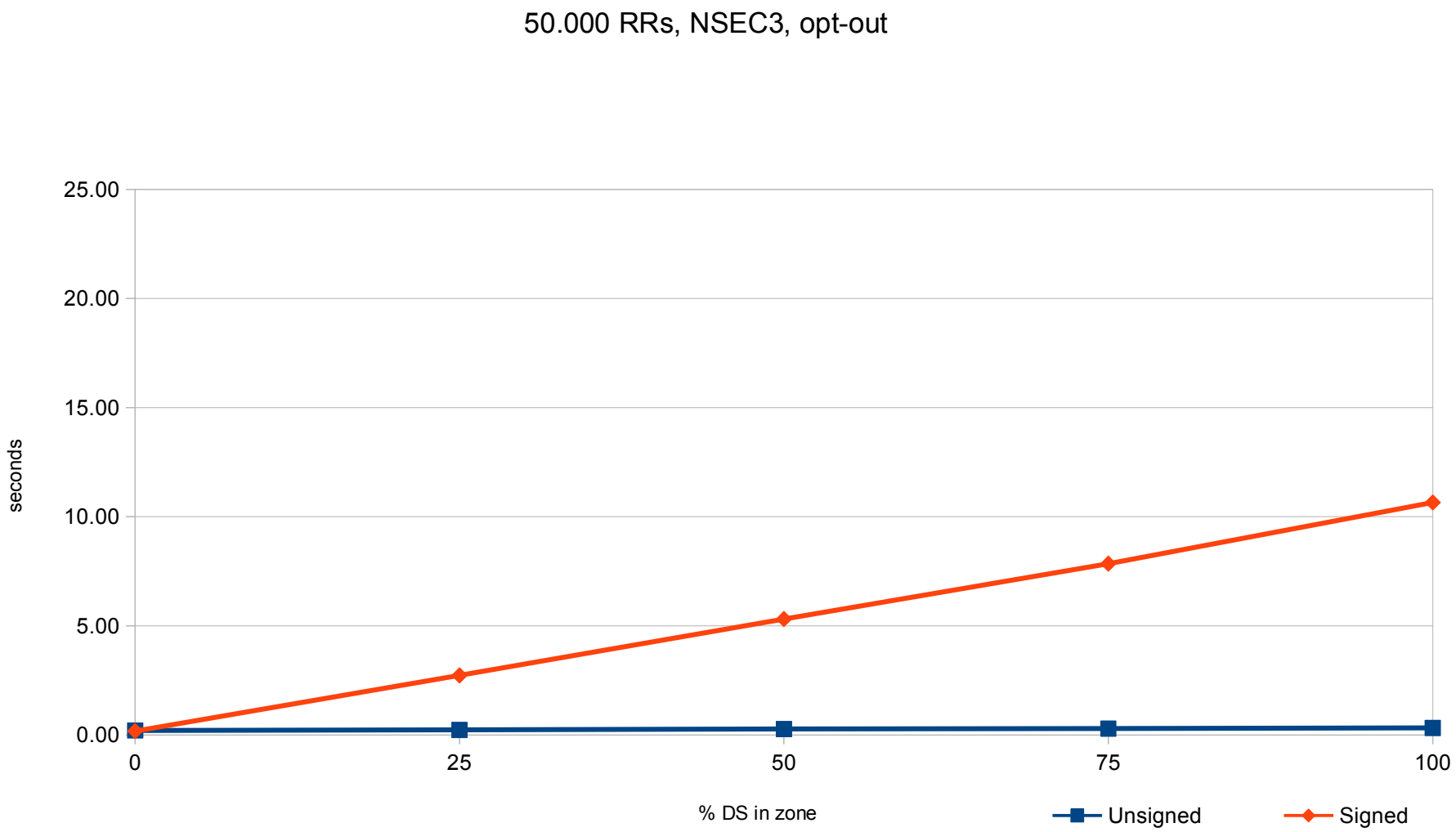
Some measurements



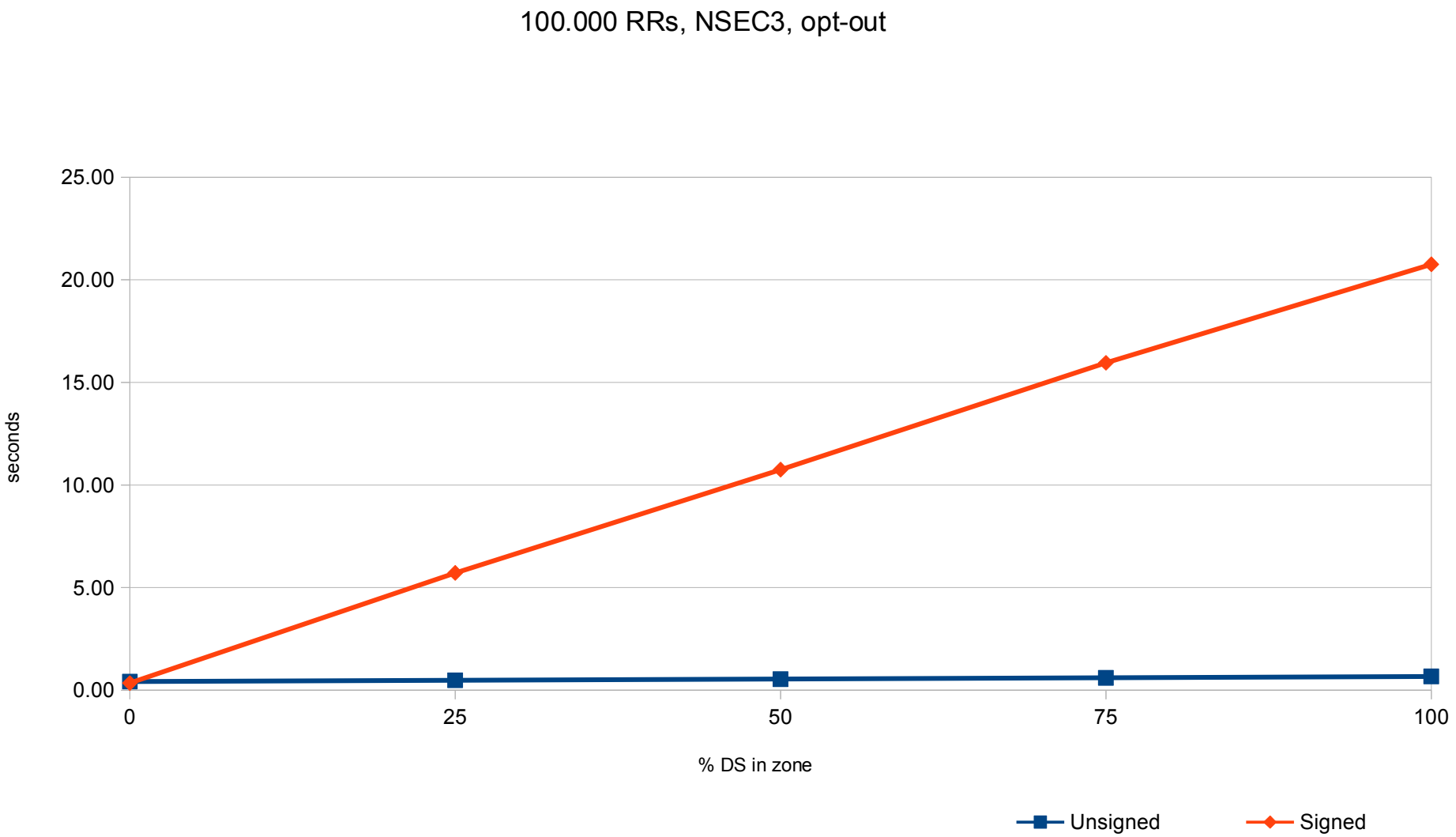
Measurements (2)



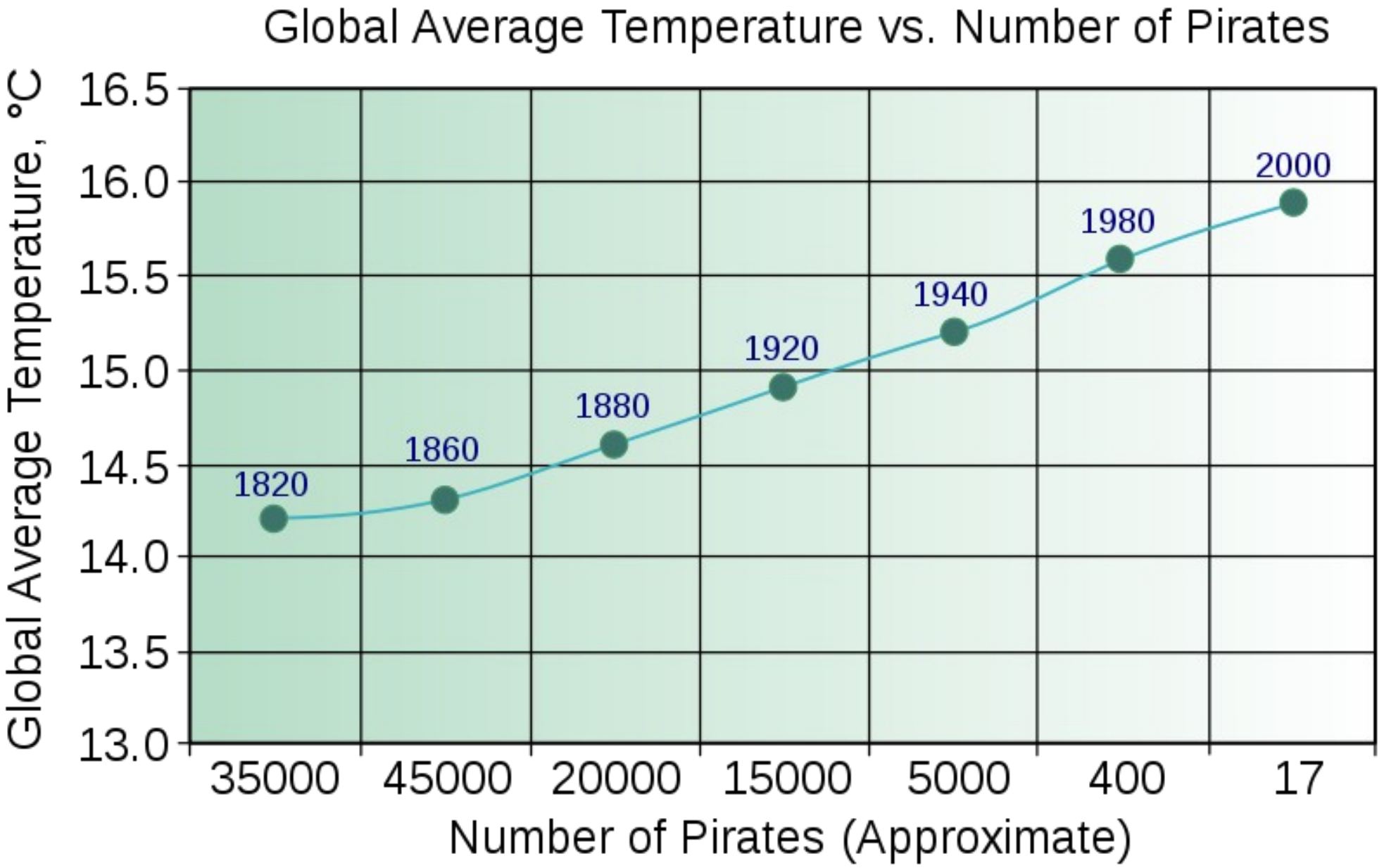
Measurements (3)



Measurements (4)



One more graph...



Analysis

- Surprising trend emerges :
  - I am not talented at making graphs in LibreOffice
  - The bigger the zone, the longer it takes to validate it (hold that thought)

Integration

- Multiple ways to use/integrate :
  - As a service check :

```
# !/bin/sh
dig @soa axfr >zone
validns zone
if [ $ ? = 0  ] ; then
  echo “all ok” ; exit 0
else
  echo “oh noes !” ; exit 1
fi
```

Integration (2)

- As a pre- & post-signing testing tool in a production pipeline :

```
 for zone in `select name from zones where HasPaid=1`
  do
    select rname, ttl, rdtype, rdata from data where zone='$zone' >$zone
    if $zone test (!NotOutOfDiskSpace && !MissingDots) then
       $zone allOK
    else
       Achtung! Broken $zone
       exit 666
    endif
    dnssec-signzone --lotsofflags $zone >$zone.signed
==> validns --moreflags $zone.signed
    if $zone.signed test (!ComputerSaysYes) then
       $zone.signed good enough for validns, my Asset is Covered
       cp $zone.signed /usr/local/dns/production/
    else
       Achtung! Broken $zone.signed
       exit 666
    endif
 done
```

Internal design – gory details

- Plain C
- No autoconf/automake/libtool
- Simple Makefile
- Minimum external dependencies
  - OpenSSL (talk about minimal...)
  - Judy arrays

Internal design (1)

- Whole zone in memory
- Lots of small allocations
  - System malloc is too slow
  - 4.3 M RRs : 11.6 secs with sysmalloc
  - Two kinds of allocations
    - Objects that persist forever
    - Objects that will be freed « soon »
  - Custom memory allocation
  - 4.3 M RRs: 8.3 secs current version

Internal design (2)

- Data structures crafted to make various traversals fast :

```
struct named_rr {
        char *name;
        void *rr_sets; // Judy array
        int line;
        char *file_name;
        uint32_t flags;
        struct named_rr *parent;
};
```

Internal design (3)

```
struct rr_set {
    struct rr* head; // linked list of RR
    struct rr* tail;
    struct named_rr *named_rr; // « up »
    int rdtype;
    int count;
};
```

Internal design (4)

```
struct rr
{
    struct rr* next; // linked list
    struct rr* prev;
    struct rr_set *rr_set; // « up »
    int     ttl;
    int rdtype; // same as in rr_set
    int line;  // for error reporting
    char *file_name; // ditto
};
```

Internal design (5)

- Poor man's OO programming

```
struct rr_a {
    struct rr rr;
    struct in_addr address;
};
struct rr_ptr {
    struct rr rr;
    char *ptrdname;
};
```

Internal design (6)

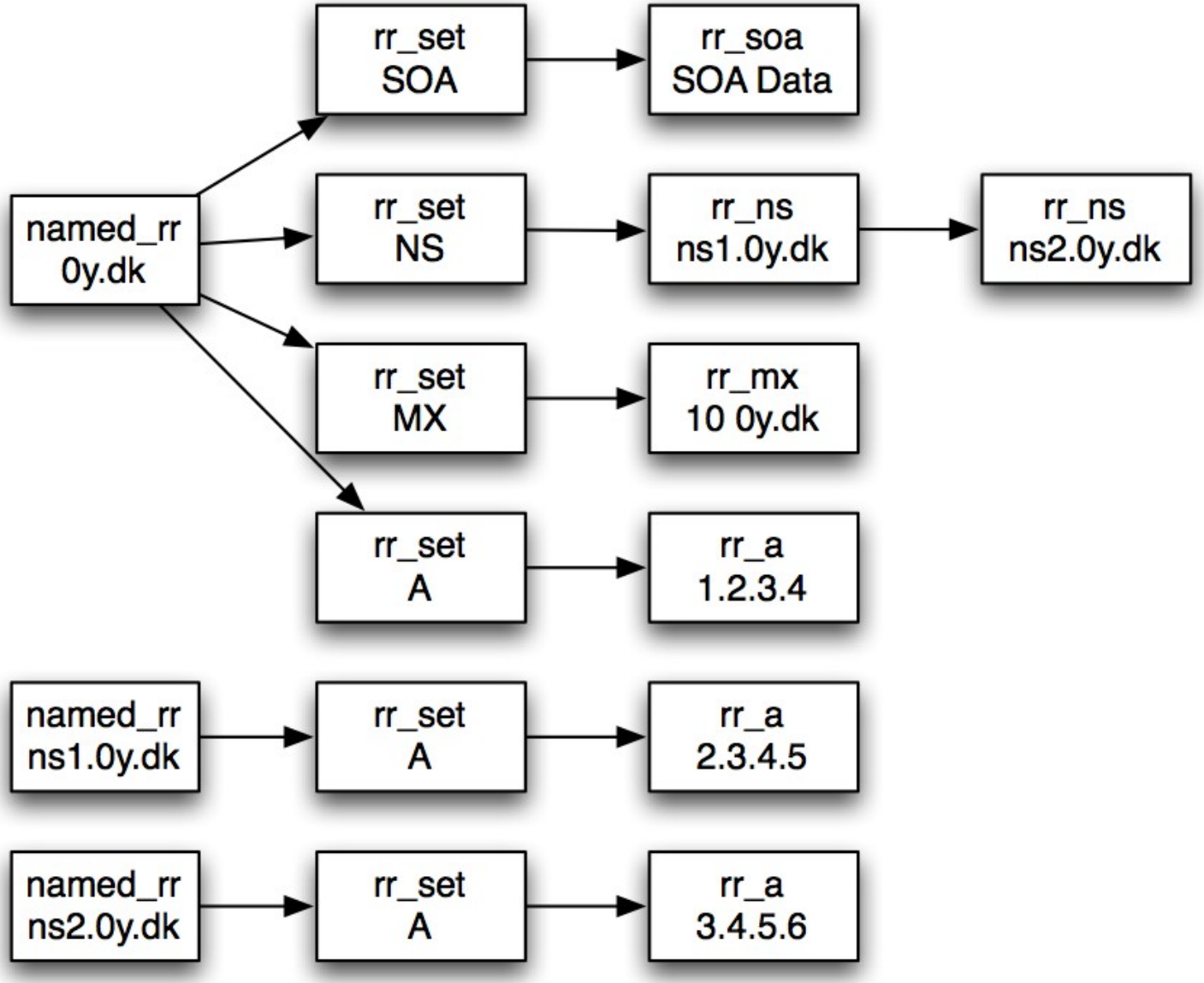
- « Virtual method tables »

```
struct rr_methods a_methods = { a_parse,
a_human, a_wirerdata, NULL, NULL };

struct rr_methods ptr_methods = { ptr_parse,
ptr_human, ptr_wirerdata, NULL, NULL };

struct rr_methods ns_methods = { ns_parse,
ns_human, ns_wirerdata, ns_validate_set,
ns_validate };
```

Internal design (7)



Who's using it, or planning to ?

- .CL – NIC Chile - in the coming month
  - 850 K RRs
- .FR – AFNIC - working on it
  - 2 M+ RRs
- .NL - SIDN – within the next month or two
  - 5 M+ Rrs
- A couple of large registrars and hosting companies

Future directions

Performance improvements

- Linear growth in validation time not cool
- Threading
  - Currently single threaded
  - Benefit from multi-core setup
- Parallelization
  - Parsing zone can't easily be parallelized
  - Checking signatures can...
- GPU acceleration

Policy improvements

- Scriptable policies using LUA
  - Measure delta / size variations in the zone
  - Conditional presence of records / counts
  - ...

Sponsors

- We'd like to thank the .FR ccTLD registry, AFNIC, for sponsoring the initial development of validns

Thanks for your time

Questions ?