Detecting latency spikes in DNS server implementation(s)

Petr Špaček

2023-02-17

https://www.isc.org
Talk structure

- Motivation: BIND bug report
- Testing
  - existing tools
  - dnsperf improvements
- Visualization
- Recommendations
After upgrading our **secondary** servers with BIND from version 9.11 to 9.16, our monitoring sometimes detects **latency spikes**. They disappear eventually.
BIND bug report
Testing latency: tools 1/2

- dnsperf 2.10
  - min/avg/max
  - at the end of test run
- resperf 2.10
  - only avg
  - per interval
Testing latency

- When Bill Gates walks into a bar, on average everyone inside becomes a billionaire.
- Histogram!
Testing latency: tools 2/2

- flamethrower 0.10.2
  - min/avg/max, per second
- shotgun 20210714
  - histogram each second, 1 ms granularity – yay!
  - histogram visualization – yay!
  - **suitable only for resolver testing** – boo-boo
    - (requires PCAP with correct query timing)
dnsperf 2.11 – new features

- Latency histogram, per second!
- `dnsperf`
  - `-S 1 # print stats every second`
  - `-O suppress=timeout`
  - `-O verbose-interval-stats`
  - `-O latency-histogram`
dnsperf 2.11 – new features

Interval Statistics: ...
Latency bucket (s): answer count

0.0000320 - 0.0000327: 30
0.032768 - 0.033791: 1
0.417792 - 0.425983: 1

- logarithmic with ~ 3 % accuracy
- clever optimizations, courtesy of Tony Finch
Interpretation

I see a small apartment...

I see you... alone...

And a lot of numbers.

Jesus that's a lot of numbers.
Visualization

- Reuse visualization from DNS Shotgun
  - Different output formats …
    - … a little hack needed …
- JSON output for dnsperfer
  - https://github.com/pSPACEK/dnsperf/tree/json_output
- dnsperfer JSON input for DNS Shotgun
  - https://gitlab.nic.cz/knot/shotgun/-/tree/oarc40
Test setup #1

- VMs in AWS ?!?
- Baseline < 400 us for 99.98% queries!
  - Long tail up to 4 ms for the rest
Echo server: latency histogram

Only 0.02% queries slower than 400us
Echo server: latency histogram

- Max across 10 test runs
- Average
- Min across 10 test runs
Echo: min/25/50/75/max boxplot
Echo server
Test setup #2

- Primary: 100 k zones + catalog zone
- Secondary: consumes the catalog
- `dnsperf` -> secondary
  - CPU load < 20 %
  - `-Q 100000 -S1 -c 256 -q 65535 -t 1 -l 60 -O json -O latency-histogram -O verbose-interval-stats -O suppress=timeout,unexpected`
Test setup #2

- static zones
Static BIND: latency histogram

**Graph Description:**
- The graph compares the response times for different slowest percentiles.
- The x-axis represents the slowest percentile, ranging from 0.001 to 100.
- The y-axis represents the response time in microseconds (μs), ranging from $10^2$ to $10^3$.
- Two lines are shown:
  - **Blue line:** labeled "echo".
  - **Orange line:** labeled "BIND v9.18.10 static".
- The blue line shows a lower and upper bound, indicated by the shaded area.
- The graph illustrates how BIND v9.18.10 static performs at different percentiles compared to the echo method.
Static BIND: latency boxplot

Latency [μs]

Test time [s]
Static BIND
Test setup #3

dnsperf

BIND secondary

BIND primary

script modifies catalog zone
BIND 9.18.10 catalog modification
BIND 9.18.10 catalog modification

![Graph showing latency over test time with color-coded lines for BIND 9.18.10 static and BIND 9.18.10 catz mod, with a client timeout indicated.]

- **Latency [us]**: Logarithmic scale from $10^2$ to $10^6$
- **Test time [s]**: Linear scale from 0 to 59

- Blue line: BIND 9.18.10 static
- Orange line: BIND 9.18.10 catz mod

**Client timeout**
BIND problem confirmed

• Catalog zone processing
  • Hash table too small
  • Degradation to linear list
    • 100k zones in catalog => 6000 items in list

• Hash table sizing fixed
  • 9.16.37
  • 9.18.11
BIND problem confirmed
BIND 9.18.11 catalog modification

![Graph showing latency over test time with different colors representing different configurations.]

- 9.18.10 static
- 9.18.10 catz mod
- 9.18.11 catz mod

Client timeout is indicated by the vertical lines.
Another problem identified

- Blind packets-to-thread assignment
  - in kernel, SO_REUSEPORT …
  - vs "long"-running operations
- A fix is in the works
Takeaway #1: outliers matter

- Averages lie
- Check raw data first
- Percentiles beyond 95% **still matter**
  - with 100 k QPS … 1 % = 1000 QPS …
Takeaway #2: timeouts

- Timeouts detected retrospectively
- For timeout = 5 seconds
  - Loss occurred in $<\text{now} - 5, \text{now}>$ interval
- Correct(ish) attribution required
Recommendations

- Outliers matter
- Attribute timeouts properly
- Upgrade, upgrade, **upgrade** ...
  - BIND 9.16.37, 9.18.11 or newer
Recommendations

- Outliers matter
- Attribute timeouts properly
- Upgrade, upgrade, **upgrade** ...
  - BIND 9.16.37, 9.18.11 or newer
Thank you!

• Main website: https://www.isc.org
• Software downloads: https://www.isc.org/download or https://downloads.isc.org
• Presentations: https://www.isc.org/presentations
• Main GitLab: https://gitlab.isc.org