nominet

Dynamic Registry Updates

John Dickinson
Senior Researcher
Nominet UK

Dynamic Registry Updates

Background

- Complete zone builds took a very long time to do and propagate. (3.5m DNs, 10m RRs, 300MB biggest zone file, 2.5 hr build/propagation). Increasing the frequency would have just increased the problem.
- BIND goes silent while reload zone after AXFR.
- Wanted to reduce the support load by people who make a mistake and cannot wait until next full build to have it corrected.
- Have seven NSs running BIND9 and service from UltraDNS for four anycasted servers (UltraDaemon).
- Can receive up to 300,000 updates per day.
- Consulted with registrars and only one objected - thought it would reduce the stability of our NSs.

Dynamic Registry Updates

System requirements

- Runs as a separate process from the main registry system, so that it can be started and stopped independently.
- Written very defensively. Uses DNS lookups to hidden primary to double-check it has worked.
- Certain errors ‘can be lived with’ even if they are not exactly correct, until operator intervention is possible.
- Must be very secure, no spoofing possible.
- Link in our existing monitoring system, which can detect error failures.

Process overview

- Registration system writes out a table of changes.
- Separate process, running as daemon, examines this table for new items, packages the updates and sends them off to hidden primary NS using DDNS.
- This process then checks hidden primary NS by doing DNS lookups to double check it has worked.
- Hidden primary notifies public primary, which then picks up the changes using IXFR.
- Public primary then notifies the secondaries and they pick up changes using IXFR.

System technology

- Written in Perl using Net::DNS. Sends changes by DDNS. Performance is very good.
- Uses TSIG to secure DDNS updates to the hidden primary.
- A single DDNS update set can contain 500 updates (uses TCP).
- DDNS has no concept of modification, so have to remove then add in the same update set.
- Sets the serial number in the SOA for each update set, otherwise NS would do it. We are using Unix time taken when the update packet is sent.

Changes to registry system

- Registry system now writes out table of changes as part of the single transaction that updates the registry.
- Need to store data on what to remove as well as what to add. (for example if NS records change then need to record which to remove and which to add).
- Negligible impact on performance
- Additional program to force details of DN into this table, in case manual processing is required.

Dynamic Registry Updates

Testing

- Used four months of data, which was 1.1 million changes. Takes about 4 hours to process.
- Built a subset of a our live network, with just three NSs, using identical hardware and software.
- Basic methodology was to replay registry updates into test system and apply changes to static zone file and compare product to second static zone file.
- Awkward corner cases discovered and then corrected.
- Finally tested failure modes: pulled out cables, switched off machines, etc.

Pre-implementation tasks

- Some months previously implemented TSIG on all NSs.
- Serial number rollover one week before as the new serial number would be lower than the first.
- Set primary to send NOTIFY and prepare secondaries to respond. (Previously only used scheduled AXFR).
- Warn external DNS provider.

BIND specifics

- Journal file set to 50MB. This holds changes as they come in. Can grow beyond 50MB quite happily.
- Every 15mins text zone files are updated from journal file. If journal file exceeds 50MB then it is truncated back to this size or smaller. Same thing happens on reload or shutdown.
- BIND never goes deaf in this process.
- Network propagation is normally less than one second across all BIND NSs for most update sets.

Ongoing support

- Scheduled tidy up of database table as update process does not delete anything, just marks it as processed.
- If a NS is down for repair then it may need to use AXFR when it comes back if change record has been overwritten on primary.
- If it all goes wrong then we can always switch back to full builds and AXFR.
- System administrators have to remember to turn update process off before restarting hidden primary.
- Review notify/IXFR structure.
- Review the size of the journal.

Some statistics

After the first 10 weeks

- Average time it takes for a change to get from the database to the nameserver constellation is:
  - 35.52 seconds
- As the process sleeps for 1 minute this is actually 5.52 seconds to do the work.
- In 10 weeks the system has processed:

– 294k	additions
– 506k	changes
– 223k	removals

- Average propagation time across NS constellation is:
  - below 1s (too quick to bother measuring)

Dynamic Registry Updates

More statistics

For the last month

- Average time it takes for a change to get from the database to the nameserver constellation is:
  - 32.48 seconds (was 35.52)
- As the process sleeps for 1 minute this is actually 2.48 seconds to do the work.
- In last month the system has processed:
  - 145k additions (was 294k for first 10 weeks)
  - 140k changes (was 506k for first 10 weeks)
  - 125k removals (was 223k for first 10 weeks)
- Average propagation time across NS constellation is still below 1s (too quick to bother measuring)

Dynamic Registry Updates

Performance

Dynamic Registry Updates

Conclusion

Very successful

- Proper development process with good testing.
- Technology was actually quite simple.
- Surprised by how fast it propagates.