Scaling DNS

Requirements

- Large domain registrar
- A few hundred thousand domains using our DNS.
  - Nearly a million domains
- Still growing
- Need fast updates on the authoritative servers
  - Especially when a new domain is added
  - Or removed

- Why use a database?
- Choosing an option
  - Stability
  - Performance
  - Support
- Options considered
  - BIND
  - BIND-DLZ
  - PowerDNS
  - MyDNS

Flat files don't always rock

- Flat files rock for system administrators
  - DNS administrators may not have root access
  - Editing flat files from the web is ... insane
- Delegating single zone management is difficult
- Edits and reloads have to be batched
- Startup times for nameserver processes

Flat files don't always rock

- Flat files cause a startup time delay on the server process
  - For BIND, this penalty is huge
    - A single host, with 113K small zones took slightly over an hour to start with flat files
    - It then served data at about 30K qps
  - PowerDNS is “optimised” for this task
    - It took over 6 minutes to parse 113K small zones and start serving data
    - This served data at about 26K qps

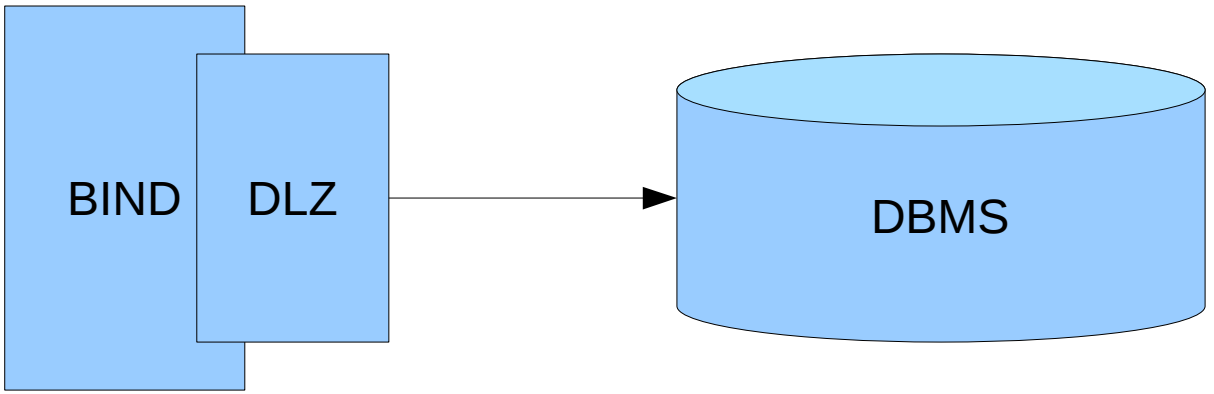
BIND

- Native BIND has “some” support.
  - One connection to the backend for each domain
  - Low performance

BIND-DLZ

- Patch to BIND
  - See http://bind-dlz.sourceforge.net/
- One/Two connections to the backend for all domains
- User defined queries, including stored procedures
- Supports MySQL, PostgreSQL, OpenLDAP, BDB

BIND-DLZ architecture



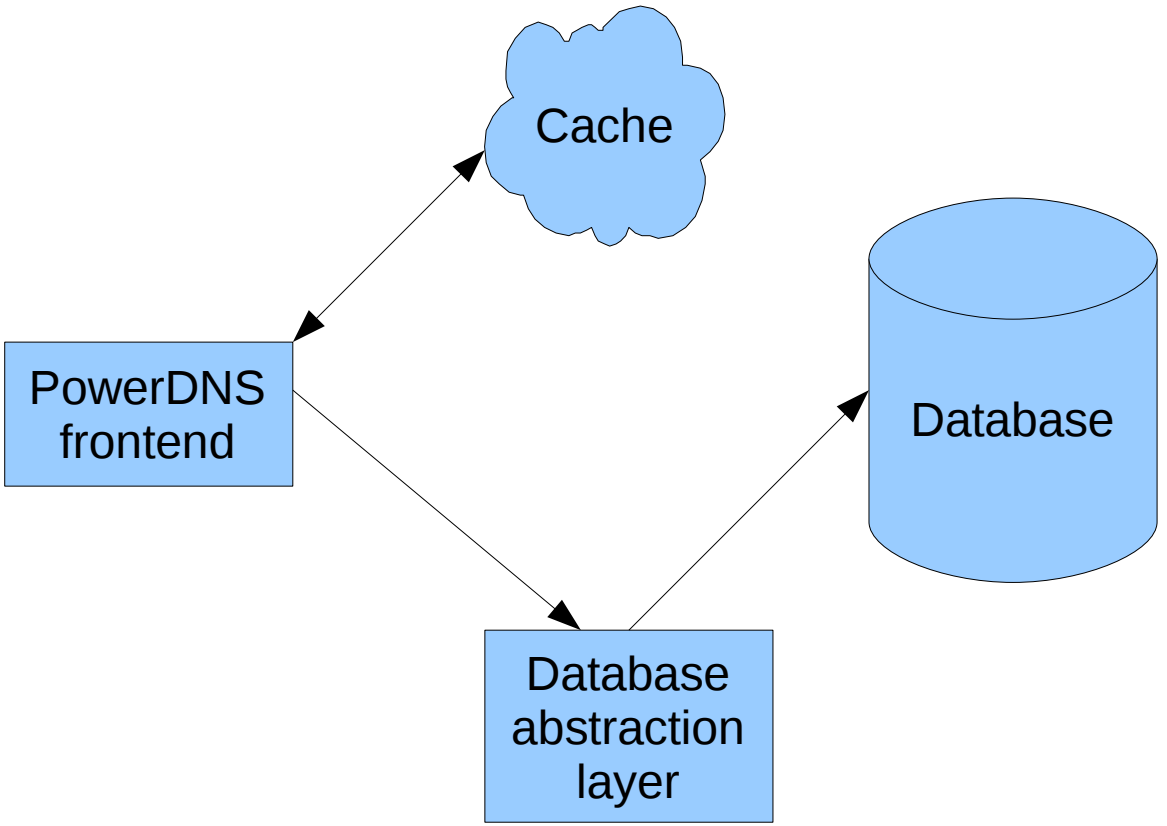
DLZ schema

- BENEFITS
  - Single connection to database
  - Single table needed to store all records
    - Keeps it simple
- DISADVANTAGES
  - No caching
  - Bad performance
  - Third party patch to BIND

PowerDNS

- DNS server written in C++
- Separate authoritative and recursive servers
- Strong focus on security

PowerDNS architecture



PowerDNS

- BENEFITS
  - One or more connections to the database
  - Two tables needed
  - Internal in-memory cache makes responses fast
  - Low memory footprint
- DISADVANTAGES
  - Cache management
  - DNSSEC (partial support, WIP) and views (no support)
  - http://www.powerdnssec.org/

MyDNS

- Not maintained – last update was 2006
- Hardcoded schema
  - We have our own.

Lies, damn lies and statistics

- Actually, performance benchmarks
- Native BIND wasn't going to be very useful to us. We didn't test it at all, except as a baseline statistic.

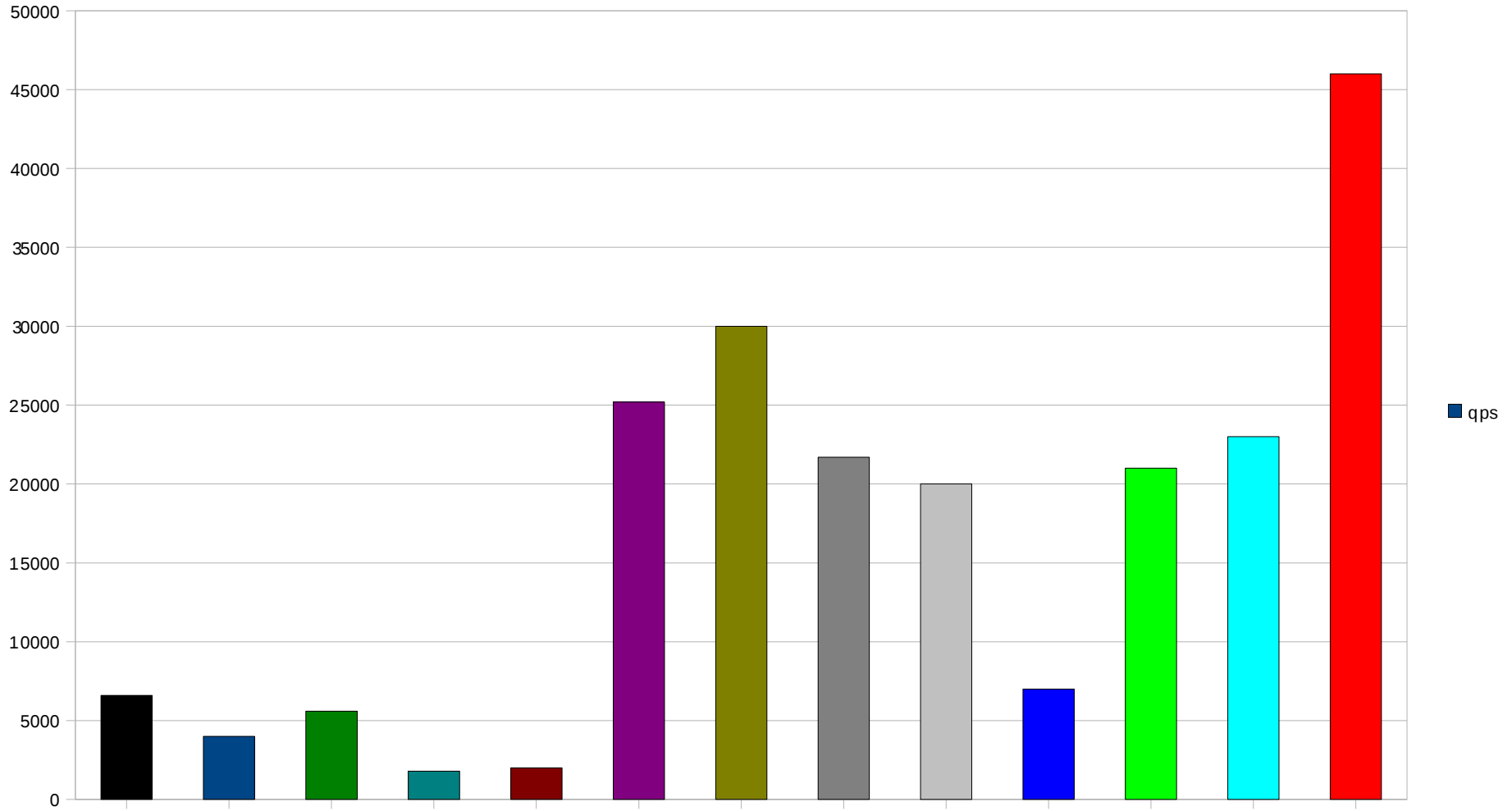
Testing

- Test infrastructure was a single host with two dual core Xeon CPUs at 2.something GHz and 16 GB of RAM with one disk, running the nameserver and the database
- The clients were three pentium boxes with 1 GB of RAM connected over a 100 Mbps network.
- We used the commandline `queryperf` tool to run queries.
  - `queryperf -q 20 -s <IP> -d <FILE>`

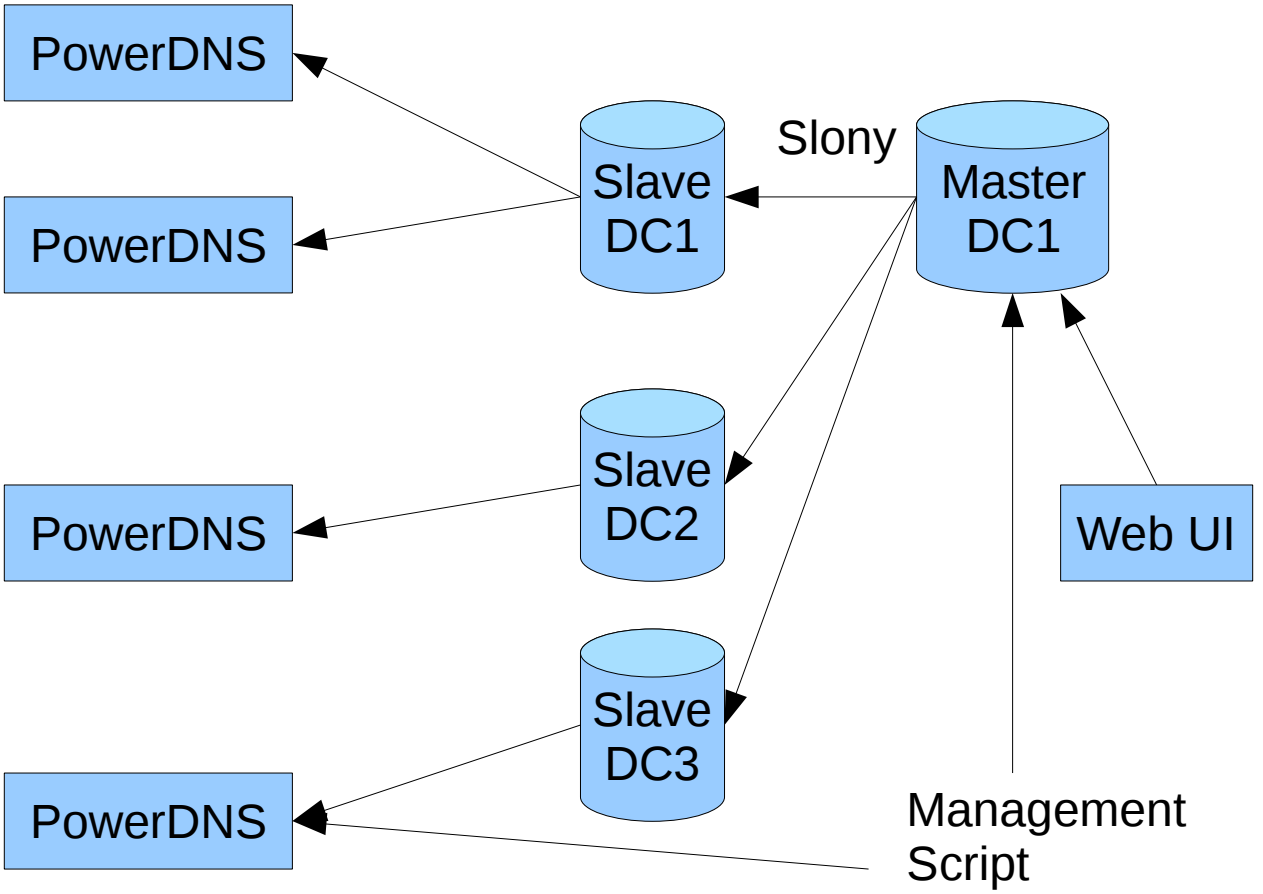
Results

BIND-DLZ with BDBHPT, 2 queryperf clients, transaction mode	4000
BIND-DLZ with BDBHPT, 2 queryperf clients, transaction mode, RAMDisk	5600
BIND-DLZ with PostgreSQL, raw queries	1800
BIND-DLZ with PostgreSQL, Stored procedures	2000
BIND-DLZ with PostgreSQL, SP, 8 DLZ threads	4800
Nominum ANS, default cache size	25200
Nominum ANS, cache size 512 MB	30000
Raw BIND	21700
PowerDNS, hash as cache, 115K domains	20000
PowerDNS, hash as cache, 3M domains	7000
PowerDNS, RBT with single threaded cache, 3M domains	21000
PowerDNS, RBT w/ single threaded cache, 10M domains	23000
PowerDNS, RBT w/ multi-threaded cache, 10M domains, 2.9.22	46000

Same numbers, as a graph



Operational architecture



Software

- Master database is PostgreSQL 8.2.x
- Read-only slaves have some tables replicated from the master database via Slony.
- Cache maintainance is a homegrown script.

In Practice

- Maximum database replication lag is 10 seconds.
- PowerDNS has peaked at ~ 10000 qps during a DdoS
  - This would have brought down the previous BIND-DLZ installations
  - We didn't notice the blip
    - Yes, monitoring needs improvement ;)

Questions?