A Comparison of DNS Server Types

Introduction

DNS, or the Domain Name System, is an integral part of how systems connect with each other to communicate on the internet. Without DNS, computers, and the people who use them, would be required to connect using only numerical addresses known as IP addresses.

Besides the obvious problem of having to remember a large number of complex numbers for simple tasks, communicating through IP addresses also causes some additional problems. Moving your website to a different hosting provider, or moving your servers to different locations would require you to inform every client of the new location.

DNS servers, the computers that together form the system that allow us to use names instead of addresses, can server many different functions, each of which can contribute to your ability to accessing servers by name.

In a previous guide we discussed some of the basic terminology and concepts of the domain name system. We will assume some familiarity with the concepts covered in that article. In this guide, we will talk about some of the different types of DNS server setups and what the advantages, use cases, and properties are of each.

The Path of a DNS Query

When a client program wants to access a server by its domain name, it must find out how to translate the domain name into an actual routable address that it can use to communicate. It needs to know this information in order to get or send information to the server.

Some applications, including most web browsers, maintain an internal cache of recent queries. This is the first place the application will check,

if it has this capability, in order to find the IP address of the domain in question. If it does not find the answer to its question here, it then asks the **system resolver** to find out what the address of the domain name is.

A **resolver** in general is any component that acts as a client-side participant in a DNS query. The system resolver is the resolving library that your operating system uses to seek out the answer for DNS queries. In general, system resolvers are usually what we consider **stub resolvers** because they are not capable of much complexity beyond searching a few static files on the system (like the `/etc/hosts` file) and forwarding requests to another resolver.
So generally, a query goes from the client application to the system resolver, where it is then passed to a DNS server that it has the address for. This DNS server is called a **recursive DNS server**. A recursive server is a DNS server that is configured to query other DNS servers until it finds the answer to the question. It will either return the answer or an error message to the client (the system resolver in this case, which will, in turn, pass it to the client application).

Recursive servers generally maintain a cache as well. It will check this cache first to see if it already has the answer to the query. If it does not, it will see if it has the address to any of the servers that control the upper level domain components. So if the request is for `www.example.com` and it cannot find that host address in its cache, it will see if it has the address of the name servers for `example.com` and if necessary, `com`. It will then send a query to the name server of most specific domain component it can find in order to query for more information.
If it does not find the address to any of these domain components, it has to start from the very top of the hierarchy by querying the **root name servers**. The root servers know the addresses of all of the TLD (top level domain) name servers which control zones for `.com`, `.net`, `.org`, etc. It will ask the root servers if it knows the address of to `www.example.com`. The root server will refer the recursive server to the name servers for the `.com` TLD.
The recursive server then follows the trail of referrals to each successive name server that has been delegated responsibility for the domain components, until it can zero in on the specific name server that has the

full answer. It puts this answer into its cache for later queries and then returns it to the client.

As you can see from this example, there are many different kinds of servers, and they each play a different role. Let's go over the specifics of the different types of DNS servers.

Functional Differences

Some of the differences between DNS servers are purely functional. Most servers that are involved with implementing DNS are specialized for certain functions. The type of DNS server you choose will largely depend on your needs and what type of problem you are hoping to solve.

Authoritative-Only DNS Servers

An authoritative-only DNS server is a server that only concerns itself with answering the queries for the zones that it is responsible for. Since it does not help resolve queries for outside zones, it is generally very fast and can handle many requests efficiently.

Authoritative-only servers have the following properties:

- **Very fast at responding to queries for zones it controls.** An authoritative-only server will have all of the information about the domain it is responsible for, or referral information for zones within the domain that have been delegated out to other name servers.
- **Will not respond to recursive queries.** The very definition of an authoritative-only server is one that does not handle recursive requests. This makes it a server only and never a client in the DNS system. Any request reaching an authoritative-only server will generally be coming from a resolver that has received a referral to it, meaning that the authoritative-only server will either have the full answer, or will be able to pass a new referral to the name server that it has delegated responsibility to.

- **Does not cache query results.** Since an authoritative-only server never queries other servers for information to resolve a request, it never has the opportunity to cache results. All of the information it knows is already in its system.

Caching DNS Server

A caching DNS server is a server that handles recursive requests from clients. Almost every DNS server that the operating system's stub resolver will contact will be a caching DNS server.

Caching servers have the advantage of answering recursive requests from clients. While authoritative-only servers may be ideal for serving specific zone information, caching DNS servers are more broadly useful from a client's perspective. They make the DNS system of the world accessible to rather dumb client interfaces.

To avoid having to take the performance hit of issuing multiple iterative request to other DNS servers every time it receives a recursive request, the server caches its results. This allows it to have access to a broad base of DNS information (the entire world's publicly accessible DNS) while handling recent requests very quickly.

A caching DNS server has the following properties:

- **Access to the entire range of public DNS data.** All zone data served by publicly accessible DNS servers hooked into the global delegation tree can be reached by a caching DNS server. It knows about the root DNS servers and can intelligently follow referrals as it receives data.
- **Ability to spoon-feed data to dumb clients.** Almost every modern operating system offloads DNS resolution to dedicated recursive servers through the use of stub resolvers. These resolving libraries simply issue a recursive request and expect to be handed back a complete answer. A caching DNS server has the exact capabilities to serve these clients. By accepting a recursive query, these servers promise to either return with an answer or a DNS error message.

- **Maintains a cache of recently requested data.** By caching the results as it collects them from other DNS servers for its client requests, a caching DNS server builds a cache for recent DNS data. Depending on how many clients use the server, how large the cache is, and how long the TTL data is on the DNS records themselves, this can drastically speed up DNS resolution in most cases.

Forwarding DNS Server

A alternative take on developing a cache for client machines is through the use of a forwarding DNS server. This approach adds an additional link in the chain of DNS resolution by implementing a forwarding server that simply passes all requests to another DNS server with recursive capabilities (such as a caching DNS server).

The advantage of this system is that it can give you the advantage of a locally accessible cache while not having to do the recursive work (which can result in additional network traffic and can take up substantial resources on high traffic servers). This can also lead to some interesting flexibility in splitting your private and public traffic by forwarding to different servers.

A forwarding DNS server has the following properties:

- **The ability to handle recursive requests without performing recursion itself.** The most fundamental property of a forwarding DNS server is that it passes requests on to another agent for resolution. The forwarding server can have minimal resources and still provide great value by leveraging its cache.
- **Provide a local cache at a closer network location.** Particularly if you do not feel up to building, maintaining, and securing a full-fledged recursive DNS solution, a forwarding server can use public recursive DNS servers. It can leverage these servers while moving the primary caching location very close to the client machines. This can decrease answer times.
- **Increases flexibility in defining local domain space.** By passing requests to different servers conditionally, a forwarding server can

ensure that internal requests are served by private servers while external requests use public DNS.

Combination Solutions

While the above solutions are built with very specific purposes in mind, it is often desirable to set up your DNS server to combine the advantages of each.

A DNS server may be configured to act as a recursive, caching server for a select number of local clients, while answering only iterative, authoritative requests from other clients. This is a common configuration because it allows you to answer global requests for your domain, while also allowing your local clients to utilize the server for recursive resolution.

While certain DNS software is specially designed to fulfill one specific role, applications like Bind are incredibly flexible and can be used as hybrid solutions. While in some cases attempting to provide too many services in a single server can lead to performance degradation, in many cases, especially in the case of small infrastructure, it makes the most sense to maintain a single, all-in-one solution.

Relational Differences

While the most apparent differences between DNS server configurations are probably functional, the relational differences are also extremely important.

Primary and Slave Servers

Given the importance of DNS in making services and entire networks accessible, most DNS servers that are authoritative for a zone will have built-in redundancy. There are various terms for the relationships between these servers, but generally, a server can either be a **master** or a **slave** in its configuration.

Both master and slave servers are authoritative for the zones they handle. The master does not have any more power over the zones than the slave. The only differentiating factor between a master and a slave server is where they read their zone files from.

A master server reads its zone files from files on the system's disk. These are usually where the zone administrator adds, edits, or transfers the original zone files.

The slave server receives the zones that it is authoritative for through a zone transfer from one of the master servers for the zone. Once it has these zones, it places them in a cache. If it has to restart, it first checks its cache to see if the zones inside are up-to-date. If not, it requests the updated information from the master server.

Servers are not relegated to only be a master or a slave for all of the zones they handle. Master or slave status is assigned on a zone-by-zone basis, so a server can be a master for some zones and a slave for others.

DNS zones usually have at least two name servers. Any zone responsible for an internet routable zone *must* have at least two name servers. Often times, many more name servers are maintained in order to spread the load and increase redundancy.

Public vs Private Servers

Often, organizations use DNS both externally and internally. However the information that should be made available in both of these spheres is often drastically different.

An organization might maintain an externally available authoritative-only DNS server to handle public DNS queries for the domains and zones that it handles. For its internal users, the organization might use a separate DNS server that contains the authoritative information that the public DNS provides, as well as additional information about internal

hosts and services. It might also provide additional features, such as recursion and caching for its internal clients.

While we mentioned the ability to have a single server handle all of these tasks in the "combination" server above, there are definite advantages to splitting the workload. In fact, maintaining completely separate servers (internal vs external) that have no knowledge of each other is often desirable. It is especially important, from a security standpoint, that the public server has no records of the private counterpart. This means not listing your private name servers with NS records in the public zone files.

There are some additional considerations to keep in mind. While it might be easier to have your public and private servers share zone data that they have in common in a traditional master-slave relationship, this can leak information about your private infrastructure into the wild.

Beyond just keeping your private servers out of the zone files themselves (essentially a publicly searchable entity), it is usually a good idea to also remove any reference to the private server in the public server's configuration files. This means removing transfer, notify, and masters configuration details so that a compromise of the public server does not mean that your internal name servers are suddenly exposed.

This means maintaining separate zone files for each, which can be extra work. However, this may be necessary for absolute separation and security.

Conclusion

You are probably aware by this stage that there is quite a bit of flexibility in choosing your DNS configuration.

Your choices will largely depend on your organization's needs and whether your main priority is to provide faster DNS resolution for a selection of clients (caching or forwarding) or to serve your domains and zones to the internet at large (authoritative servers). Combination

approaches are common and, in the end, both sides of the resolution process need to be accounted for.

In our next guides, we will demonstrate how to get started with some of these configurations. We will begin by teaching how to set up a caching or forwarding server. Later, we will cover how to serve your domains by setting up a pair of authoritative-only DNS servers.

Source : https://www.digitalocean.com/community/tutorials/a-comparison-of-dns-server-types-how-to-choose-the-right-dns-configuration