SUN BLUEPRINTS APPROVED

Solaris™ Directory Services: Past, Present and Future

By Tom Bialaski - Enterprise Engineering

Sun BluePrints™ OnLine - October 1999

http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 8xx-xxxx-xx
Revision 01, October 1999

Solaris™ Directory Services: Past, Present and Future

Synopsis

Tom Bialaski looks at the high availability features of currently supported Solaris™ operating environment directory services (NIS, NIS+, DNS) and contrasts them with LDAP's high availability features.

Introduction

This is the first in a series of articles that view the changing role directory services play in the Solaris™ operating environment and how to enable availability of these services. Directory, or *naming*, services have long been a critical component of the Solaris operating environment distributed computing model. With the computer industry's adoption of Lightweight Directory Access Protocol (LDAP) based services, their role is becoming even more critical. Applications, as well as the operating systems they run on, will become increasingly dependent on the availability of directory services to function properly in the future.

To gain a better understanding of how high availability (HA) features in LDAP based servers, such as the iPlanet Directory Server work, we will take a look at the HA features found in the Solaris operating environment naming services NIS and NIS+, which are widely deployed today. Those features are then contrasted with the ones found in the LDAP based iPlanet Directory Server. We will also see how Solaris operating environment clients interact with these different directory services.

What Do Directory (or Naming) Services Do?

A naming service organizes and names objects so their clients can retrieve information about those objects easily. A directory service is a type of naming service that is distinguished by how the objects are stored. These objects are stored as directory entries that reside in a directory tree. Since the distinction between the two terms is not important for the purpose of this article, I will use them interchangeably.

Perhaps the most critical function a naming service performs is to authenticate a user. To gain access to a system or an application, the user enters a unique user ID along with a password associated with that ID. The program performing the authentication then contacts a naming service to retrieve the password that was stored for that user so a comparison can be made. If the directory service is unavailable, the user cannot gain access to the system or application.

How are Naming Services Made Highly Available?

Two things increase the availability of naming services. First, the information contained in them must be replicated on different servers, and second, the client accessing a naming service must be able switch to an alternate server if the primary one fails. The frequency at which data is replicated will determine how current the data is, so attention to when and how replication is performed is important.

Another critical aspect of availability is how often updates to the information in the naming service typically occur. This is important since Solaris naming services are designed using a single master model. This means that updates can be performed only on the master server. If the master server is down, then no updates can occur until it is back on line.

Historically, the information stored in Solaris naming services has been fairly static requiring only occasional updates, such as when user accounts are created or removed and when new computers are attached to the network. If the master server is off-line for a few hours while repairs are being made, it is usually acceptable. However, as the role of directory services changes, the ability to switch master servers quickly becomes critical.

There are basically two approaches to switching master servers: one approach is to promote a slave (or replica) server to a master; the second approach is to create a cluster consisting of an active master and standby server. In the event of a failure, the standby server assumes the role of master server. A future article will address clustering techniques, so that topic will not be discussed here. Instead we will focus on how to change the role of a server.

Replication Techniques

An understanding of how directory data is propagated during replication is important to properly configure directory services for maximum availability. We will now take a look at how changes to directory data are propagated from masters to slaves in the NIS/NIS+ environment and the iPlanet Directory Server environment.

NIS/NIS+

NIS stores information in binary files called *maps* which are generated from text files. The creation of these maps always takes place on the NIS master server, then are propagated to the NIS slave servers. Updates are performed by first editing text files, then generating a new version of the corresponding map. The entire new map is then transferred to the slave servers, even though only a small portion of the data may have changed.

Propagation of the new maps can be initiated either by the master server, which pushes the maps or by the slave servers, which request a copy of the new map. Since some NIS maps can become quite large, it is common practice to update and propagate maps only when the network bandwidth utilization is low.

NIS+ stores information in tables which are updated on the NIS+ master server. The updates are performed as transactions which get recorded in a transaction log. The objects on the NIS+ master server get updated immediately, but updates to NIS+ replicas are delayed for approximately 2 minutes to see if additional updates are coming. This form of batching reduces the amount of network traffic.

When the NIS+ master server is ready to send out updates, a message is sent to all its replicas. The replicas respond to the master with the time of their last updates. This time is compared with the timestamps kept in the transaction log, so only new updates are sent. Since only incremental changes are sent, updates can happen frequently without adversely impacting network bandwidth.

iPlanet(TM) Directory Server

The iPlanet Directory Server stores information in a directory information tree (DIT). The whole tree or a section of the tree can be replicated on another server. The server where updates are made is called the *supplier* server and the server that gets a copy of the supplier server's data is called the *consumer* server. In a broad sense, this is similar to the master/slave relationship in NIS and the master/replica relationship of NIS+. However, there are some important differences. For example, an iPlanet Directory Server can be both a supplier and consumer at the same time.

Like NIS+, updates are written to a *change log* before they are propagated to the consumer servers. Unlike NIS+, supplier servers do not automatically send out a message to consumers informing them that updates have been made. Instead a replication agreement is established between the supplier and the consumer. There are two types of agreements: 1) Supplier Initiated Replication (SIR), and 2) Consumer Initiated Replication (CIR). In most cases SIR is the preferred method. CIR was implemented for cases where directory servers are connected using dial-up lines, which is rare these days.

The SIR agreement specifies a policy on how changes should be propagated. If immediate changes are not required, then updates can be deferred to specific scheduled times. Replication can also be specified to take place after a certain number of updates have taken place. Like NIS+, the iPlanet Directory Server change log contains timestamps, which are compared with the time the consumer last received updates.

Changing the Server Role

If the master server fails, clients can still access directory data from a slave or replica server, but updates cannot be performed. Since the data typically contained in NIS/NIS+ usually does not require immediate updates, taking the master server off-line to perform repairs does not have great impact. However, if for some reason you cannot wait for repairs or replacement of a master server, a slave or replica can be promoted to a master.

Updates to NIS/NIS+ are usually performed by a systems administrator, but this is not always the case with LDAP enabled web based applications. These applications must be available 24 hours a day so it is not always feasible to wait for a systems administrator to repair a broken master server. Some sort of automatic failover is required. Techniques for performing automatic failover will be discussed in a future article, but for now we will look at the manual steps required.

NIS / NIS+

The master, or owner, of a NIS map is determined by where the map was created. Each map contains a key-value pair which specifies its owner. Any NIS server can be the master of a map, simply by generating one and propagating it to the other servers. However, to reduce confusion, all NIS maps are typically mastered on one server.

In the event that the current master is not operational and you need to update a map, the ownership can be transferred to another server. The steps to do this are:

1. Get a copy of the text file used to create the map. If the file is not available, a new one can be generated using the `ypcat -k` command.

2. Modify the `/etc/yp/Makefile` file on the slave server to include an entry for the map you want to create.

3. Run the `ypmake` command to create the new NIS map from the text file.

4. Run `ypxfr` on each NIS slave server specifying the new master server as the host.

NIS+ master servers *own* all the tables in their domain or subdomain. The ownership is established by the flag used when the `nismkdir` is run to create the NIS+ directory on the server. The `-m` flag specifies a master server while the `-s` flag specifies a replica server. If the `-m` flag is run specifying an existing NIS+ domain, the new server becomes the master, relegating the old master to a replica.

iPlanet(TM) Directory Server

Master and replicas are identified by the presence of a particular attribute called `copiedFrom` in the root entry of the DIT. If a server contains this attribute, it is a replica and the value of the attribute identifies the master server. If the server does not contain the attribute, then it is the master.

To promote a replica to a master you simply delete the `copiedFrom` attribute from the root entry. The following is an example:

```
# ./ldapmodify -h newhost  -D “cn=supplier” -w “password”

dn:o=sun

changetype: modify

delete: copiedFrom

modifying entry o=sun
```

How Naming Service Clients Handle Failover

NIS / NIS+ Clients

NIS and NIS+ clients can locate servers either by sending a broadcast and waiting for a server to respond or by looking up a server name in a list. The server list method is preferable since it is the most secure because any server can respond to a broadcast whether it is a legitimate one or not.

The list of servers is created when the client is first initialized. This list is maintained in the `/var/yp/bindings/ypdomain/ypservers` under NIS and in the NIS_COLD_START file under NIS+. Because the list is searched in order, always list the servers nearest the client first. In case the client's primary server fails, a server closest to the network the client is on will be picked next.

LDAP Clients

How LDAP clients locate their servers depends on how the application is written. The Netscape Communicator, for example, only accepts a single server entry. If that server is unavailable, it will not automatically failover to a different one. The LDAP client that is part of the Solaris 8 operating environment has a facility for locating a primary LDAP server and automatically switching over to alternate ones. Details on this implementation will be discussed in a future article.

How Naming Services Interoperate

With several Solaris naming services to choose from (NIS, NIS+, and DNS are currently available and LDAP will be available in the next release) how does a naming service client know where to look? The answer is the Solaris Name Service Switch.

The Solaris `/etc/nsswitch.conf` file lets the client choose which naming service(s) to use and in which order to search them. The following is an example of what this file looks like:

```
hosts:      nis [NOTFOUND=return] files
networks:   nis [NOTFOUND=return] files
protocols:  nis [NOTFOUND=return] files
rpc:        nis [NOTFOUND=return] files
ethers:     nis [NOTFOUND=return] files
netmasks:   nis [NOTFOUND=return] files
bootparams: nis [NOTFOUND=return] files
publickey:  nis [NOTFOUND=return] files
```

In this example, the system is set up to search the NIS naming service first and if it is not available, the files `/etc` are searched. The nice thing about using this configuration file is that a loss of the NIS server will not prevent a user from logging into the client as long as there is an entry in `/etc/passwd` and `/etc/shadow`.

Although the Solaris Name Service Switch can be used to specify a backup naming service, its main benefit is to provide a naming service independent interface to applications. Applications do not have to be rewritten to take advantage of a new naming service such as LDAP. More detail on the Solaris 8 operating environment implementation of the new `ldap` entry in `nsswitch.conf` will be presented in a future article.

Conclusion

Creating a highly available naming service environment requires understanding of how naming servers replicate their information and how clients interact with the servers. How and where the data is mastered is also important since loss of a master server could mean that data cannot be updated. While this might not be an issue with the way Solaris naming services are being used today, deployment of LDAP-based naming services in the future could change that.

Author's Bio: Tom Bialaski

Tom joined Sun Microsystems in 1984 as a Systems Engineer and has been providing network computing solutions to customers since then. He is currently a PC interoperability specialist and has recently received his MCSE certification from Microsoft.