NEXTSTEP In Focus, Summer 1993 (Volume 3, Issue 3).

Propagating Updates

Alan M. Marcum and Garth Snyder

The only database that a NetInfo client can modify is the master database. Clones in a domain can only be read, not written, by clients. If a writing client happens to be connected to a clone server, it reconnects to the master before it writes new information.

So, how do modifications to the master's database reach the clones? Through an operation called *update propagation*. This article explains how update propagation works, including its performance optimizations and how it handles tricky updates.

WHEN THE DATABASE CHANGES

When the master executes a write request, it makes the change in its database. Then it sends the update along to all the clones of the domain. This update propagation is automatic. System and network administrators don't need to manually intervene or tune the system.

Yet, although the update process sounds simple, it has subtleties that are interesting to explore.

What is sent to the clones

In general there are a few different ways to inform copies of any database, including clones in NetInfo, about changes to the database. One possibility is to send the entire database to the copies. Another way is to send only the data that changed. A third is to send a portion of the database, smaller than the entire database but greater than only the changed information. And a final option is to send some sort of transaction log that allows the copies to reconstruct the changes.

NetInfo usually uses a modified version of the final option of propagating updatesÐthe NetInfo master server sends the transactions to the clones. For example, if the client that made the change used an **NI_WRITE** operation, which writes all of a directory's properties, the master also uses an **NI_WRITE** operation. If the client used an **NI_WRITEPROP** operation, which writes only one property, so does the master.

(Under special circumstances, the master sends the entire database to individual clones. See ªWhen a Clone Misses an Update,º later in this article)

Updating immediately

When the master receives and executes a write request, it notifies the clones of the change to the domain immediately. This quick response helps ensure that all copies of the database are in sync with the master with minimal effort from an administrator, and with minimal chance for errors. The master sends the update to each clone in turn.

KEEPING UPDATE PERFORMANCE HIGH

When there are many clones or many changes, the master server uses multi-threaded propagation and update coalescing to keep update propagation performance high.

Multi-threaded update propagation

If there are lots of updates, as is typically the case with a large network, the master server takes steps to decrease the delay between when the master gets the write request from the client and when it begins to propagate the update. When a client makes an update, the master begins propagating the update in a separate thread. If another update request arrives while the first is being propagated, the master starts a new thread to propagate the second update.

Multi-threaded update propagation is new in Release 3. In both Release 3.0 and Release 3.1, a single master can handle up to five update threads at one time.

Update coalescing

Another performance optimization NetInfo uses is *update coalescing*. When a master starts an update thread, it combines all pending updatesÐthose that no threads are currently propagatingÐand sends them to each clone using a single connection. Multiple messages are sent through the connectionÐone message per update transaction. However, establishing the connection only once requires less overhead than sending each update through a separate connection.

When an update arrives while another update is being propagated, it isn't coalesced with the other update. Once an update thread begins running, the updates it propagates don't change. But, if all five update threads are running and a new update arrives, the new update waits in the work queue until a thread becomes

available. Once a thread is available, the master coalesces all the updates waiting in the work queue, so they are all handled by the freed thread.

Update coalescing improves update performance in networks whose domains change frequently. There's a small price to pay, though: there's a two second delay between when the master receives a write request and when it begins propagating the change. This delay allows the master to coalesce changes whenever possible.

Update coalescing was also first implemented in Release 3.

WHEN A CLONE MISSES AN UPDATE

What if a clone is down when an update is sent, or if the master cannot reach the clone because of a network failure? Sure, the master tries hard: it waits for a 60-second timeout period before giving up on sending an update to a clone. Still, failures can occur.

As an example, assume a network problem causes a clone to be inaccessible to the master for a while, although the clone server's computer continues running. While the clone is inaccessible, a client process makes a change to the domain. When the master propagates the update, the inaccessible clone doesn't receive it.

So, when the network problems are fixed, how will the clone's database be updated? Once the clone comes up, it'll begin getting updates, but what about updates sent while the clone isn't reachable?

Updates at startup time

When either a clone or the master starts up or is restarted, the clone's database is

updated. When the master starts up, it tells all the clones that it has started, and sends its database checksum with the notice. If a clone is out of date, the clone requests a new copy of the current database.

Similarly, when a clone starts up it checks in with the master, sending the clone's checksum. Again, if the two checksums differ, the master sends the clone the entire database.

Updates after startup

It's possible, though, that both the master and clone could run for a long time without being restarted. This would seem to allow the clone's database to remain obsolete for quite a while.

However, every 30 minutes the master and the clones verify that all copies of the database are current, by verifying that the clones' checksums agree with the master's. So, a clone can't stay out of date for long.

Updates on the way

There's one instance, though, when the master won't send an out of date clone a copy of the database. If an update is being propagated that hasn't yet been sent to the clone, the master tells the clone that an update propagation is in progress. The clone **netinfod** then sends a message to its syslog:

```
transfer failed at Master server is busy: id=-1
```

The clone will be current once it receives the update.