Sundisk Product Plan

Bob Lyon

1. Introduction

Sundisk is the first release of software for diskless Suns supported entirely by the NFS protocol. This differs from previous diskless support because previous releases depended on Sun's proprietary ND protocol to support diskless Suns.

Sundisk is is the version that allows a Sun server to support diskless Suns. **NFSDisk** is a more general product patterned after **sundisk**.

The first release of the **sundisk** product is piggy-backed on top of existing Sun software such that "major release" issues (like setup) do not affect it initial deployment.

The ND protocol will be eliminated from Sun products for the following reasons:

- ND administration is a nightmare. System administrators must guess how much (and in what manner) the disk utilization of each ND client; this is effectively impossible to do. Then, once numbers are determined they must be crypticly added to a "nd.local" file. The maintenance of the nd.local file is error prone.
- ND cannot be tuned. Because the ND server deals with disk partitions and not file systems, client partitions are kept on one disk and NFS based systems are kept on another. Because of this, disk traffic cannot be efficiently balanced among the disks on a server.
- ND servers can barely support heterogeneity. Any particular ND server can only support two "pub" partitions. As delivered in 3.0, one partition serves Sun2 clients and the other serves Sun3 clients. These servers will not be able to also serve new architectures like Sun4's or Roadrunners.
- The ND protocol and code is closed and proprietary; neither are well understood by Sun engineers. Furthermore, the ND code and administration tools (like `fsck`) are not portable. This means that Sun allies (such as Convex or Gould) cannot exclusively serve diskless Suns from their iron. Replacing ND with NFS will allow this happen. (However, most non-Sun machines cannot keep up with Sun workstation packet rates, thus they may deliver worse than expected performance.)
- Memory usage and reboot performance are improved. Removal of the ND driver *and* the UNIX file system code saves over 60K bytes in the diskless machine's kernel. Since the diskless machine has no UNIX file system, there is nothing to check when the machine is rebooted.

2. Product description

The product is composed of two items.

The first item is some number of magnetic (*tar*) tapes containing all software necessary to run diskless Suns. Sub-products of **sundisk** will be distinguished (and ordered) by the architecture type (Sun2, Sun3, ...) of the client machines that will be served. The product will not depend on the machine type of the server.

The second item is a "Principles of Diskless Operation" manual that describes each phase of machine bootstrap up to multi-user UNIX. The document will include bootstrap protocols, descriptions of how the diskless machines use those protocols, and descriptions of how the Sun

server sides work.

2.1. Server description

The product will be easily installable on any server running Sun Release 3.2. Note that only Sun hardware runs release 3.2 or later. However, since **sundisk** will server as the **NFSDisk** prototype for non-Sun servers, any new executable that is required on the server will be delivered as source. This will give non-Sun NFS (UNIX) vendors a chance at successfully installing the product on their iron. The following is assumed about the target server:

- Clients to be served are installed from scratch. Existing diskless clients will have to be removed and re-installed. Client files can make the transition with existing Sun OS utilities.
- There is enough disk space on the server. (A more concise statement with regard to disk space must be made.) This will typically imply that **sundisk** cannot be installed on an existing ND server without doing either one of (a) adding yet another disk, or (b) "converting" the existing client base to **sundisk** and reaping the ND disk space.

2.2. Client description

All client software will be based upon the Sun 3.2 release. The `/boot` and `/vmunix` (and some utilities like `mount`) will be dramatically different from 3.2, but all user commands should be the same.

Human users of the diskless machines should see no difference except that some files will be in new directories. See file system reorganization section below.

2.3. What is missing

In order to serve diskless clients, servers must allow root access to the clients' root file systems. The ultimate **sundisk** product will be based upon secure RPC authentication, but the original release will be based upon the existing (insecure) UNIX style RPC authentication.

3. Performance

Sundisk performance will be the same as (or better than) ND/NFS based performance for diskless clients in 3.2. (This statement is very risky since 3.2 is nowhere near to shipping and we are working at improving its performance. Equating **sundisk** performance to 3.0 performance is a very safe bet.)

The NFS group will attempt to increase the number of diskless clients per server in the **sundisk** release, though we cannot commit to this. We will commit to not decreasing this ratio.

3.1. Performance Tasks

NFS will never be as efficient as ND in per operation comparisons. We must rely on NFS architecture beating out ND architecture. Here are some ideas the NFS group is pursuing (get more from Dan):

- The server only flushes an updated file's inode if the length has changed. This makes swapping to NFS based files as efficient (as measured by clock time) as swapping to ND based disk partitions.
- RAM based NFS server for `/tmp`. Each diskless client will (optionally) run a RAM NFS server for its own `/tmp` directory. This will keep $7 - 20\%$ of all file operations from reaching shared NFS servers.
- Arbitrarily long lived NFS attributes cache. This feature is selectable on a per file system basis at mount time. By making remote file attributes long lived (infinite), a NFS client need not worry about validating its cache, thus providing performance similar to ND's. This feature can be applied to a client's root (/) mount point (since each machine keep a separate and private root, though the file system space is shared among all clients), and to

many read-only mounted file systems (this is similar to ND's `/pub` partition).

4. File System Reorganization

In order to make Sun client software (and other files) easily installable on server machines, the files need to be reorganized into a smaller collection of subtrees. The reason why UNIX files are not organized in an acceptable fashion is historical.

A major reason for the current organization is that UNIX should have a small root file system that easily fsck's and contains barely enough tools to compile "your" (a guru's) way out of any booting problem. Then once the root file system is up, other mounts can take place to turn the machine into a useful system. Within this file system, administration commands (like `fsck`) were put in the etc directory instead of the `/bin` directory so as to not confuse the normal users!

Furthermore, various files (per machine databases) found their way into a plethora of directories because some software writer thought it made sense at the time (usually due to disk space considerations). Examples of such file are `/etc/passwd`, `/usr/lib/crontab`, and `/usr/adm`.

Since client machines have no disks, the historical reasons are meaningless and only serve to confound the installation process.

The client machine will do only one mount (`/pub`) to obtain a complete environment. The `/pub` will contain all files that are found in ND's `/pub` plus many others like `/etc` executables, `/usr/bin`, `/usr/ucb`, `/usr/etc`, `/usr/games`, `/usr/local`, `/usr/lib` and `/usr/include`.

The `/etc` directory will be left for the (private) per machine databases such as `passwd`, `host`, `crontab`, and `adm`. Also, the `/usr` directory will (most likely) retrograde back to a private directory within the `/` file system, rather than being an NFS mount point; thus, `/private` and all the disgusting symbolic links to it disappear.

Compatibility issues are discussed in a section below.

4.1. NB: Server Implications

This reorganization plan does not attempt to share clients files with the server. This is a departure from the current diskless/ND/NFS strategy. E.g. the server coming up single user may now rely upon it own private copy of bootstrapping utilities and then mount the clients' `/pub` as its own.

This strategy may seem more wasteful due to redundant copies of bootstrap utilities. However, this new strategy allow sharing of clients' `/etc` executables (about 1.5 megabytes) and the "wasted" disk space is recovered by serving as few as three clients from the server.

5. (in)Compatibility

The client software will run all 3.2 executables except for the ones that depend upon the existence of a UNIX file system.

The file system re-organization is not completely compatible. Most compatibility is easy achieved at the directory level via symbolic links, i.e. `/usr/ucb -> /pub/ucb`. However, compatibility with `/etc` executables will not always be supplied. Clearly `/etc/mount -> /pub/etc/mount` is desirable, but various deamons' names need not have the symbolic links. This fits our philosophy that "if you are smart enough to use the command, you are smart enough to find its new location." Furthermore, subtle incompatibilities may be found as we convert to the reorganized file system.

The (optional, yet encouraged) use of the client-based RAM NFS server for each client's `/tmp` introduces the incompatibility of temporary files actually disappearing when the client machine reboots. Applications that wish to save temporary files across reboots should put those files in other directories like `/usr/tmp`. The editor, `vi` is an example of such an application.

Sundisk servers must allow root access to their clients. This means that super user access to those file systems will work once again (this has been "broken" since release 2.0). However, it also means that malicious users could cause severe damage to those file systems due to the insecurity of UNIX authentication.

Sun2's which have ND built into their PROMs will be supported via a user-level ND server process which only serves client boot blocks. The boot blocks will contain new code which follows the Sun3 PROM bootstrap conventions.

6. Milestones

Task	Scheduled	Actual
Add TFTP and RARP to Sun3 PROM	Release 3.0	done
NFS bootstrap `/vmunix`	June 9, 1986	done
Boot assist server	June 9, 1986	done
Write user level ND boot block server for Sun2's	Sept 1, 1986	
Release 3.2 stabilizes	Sept 15, 1986	
Vnode and NFS-ize `/boot`	Sept 15, 1986	
File System Reorganization	Oct 1, 1986	
Begin Alpha Test	Oct 15, 1986	
Begin Beta Test	Nov 15, 1986	
First Customer Ship	Feb 15, 1986	

The product is expected to be very stable by the end of alpha test. The extended beta test should allow the NFS consulting group to gain feedback on non-Sun servers.

Although not part of the release, support for Sunrise and/or Roadrunner could also begin during the beta test period.

7. Release issues

Clearly kernel code will have to be forked from mainline 3.x releases. Also a very small number of utilities (like `mount`) will also have to forked.

The objective is to not burden the Release Engineering group with another major release, but to still have a releasable and supportable product. (I need to work with Paula to nail this down.)

Beta sites need to be selected. Unlike other beta programs, the site should include both Sun and non-Sun servers.