Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
FAX 415 969-9131

For U.S. Sales Office locations, call:
800 821-4643
In California:
800 821-4642

Australia: (02) 413 2666
Belgium: 32-2-759 5925
Canada: 416 477-6745
Finland: 358-0-502 27 00
France: (1) 30 67 50 00
Germany: (0) 89-46 00 8-0
Hong Kong: 852 802 4188
Italy: 039 60551
Japan: (03) 221-7021
Korea: 822-563-8700
Latin America: 415 688-9464
The Netherlands: 033 501234
New Zealand: (04) 499 2344
Nordic Countries: +46 (0) 8 623 90 00
PRC: 861-831-5568
Singapore: 224 3388
Spain: (91) 5551648
Switzerland: (1) 825 71 11
Taiwan: 2-514-0567
UK: 0276 20444

Elsewhere in the world,
call Corporate Headquarters:
415 960-1300
Intercontinental Sales: 415 688-9000

CacheFS enhances performance in environments where repeated access to the same data is frequent. Note that in the case of a cold cache, client performance is roughly NFS client performance offset by some constant value. The constant value is the cost of cache population. The load placed on the server by clients with cold caches is roughly the same as the load imposed by plain NFS (uncached) clients. CacheFS clients should never load a server more than NFS clients would.

CacheFS Feature Summary

- CacheFS requires no modifications to existing front or back file systems. In Solaris 2.3 and subsequent releases, NFS and HSFS should be considered as back file system candidates. In Solaris 2.3, UFS should be used as a front file system. In Solaris 2.4, TMPFS may also be used as a front file system (although TMPFS cannot be used as a nonvolatile cache).
- There is no special disk partitioning required to make a cache. A cache can be created as a sub directory of an existing file system, or an entire file system may be dedicated to caching. We recommend dedicating an entire file system to caching for simplicity.
- All file types (regular files, directories, symlinks, special files, etc) are cached.
- By default, CacheFS will maintain consistency with the back file system using a consistency checking model like that of NFS (polling for changes in file attributes).
- Cache resources are managed transparently. There is no need for user intervention in cases where a cache fills. When a cache fills, cached files are discarded (on a least recently used basis) until sufficient space is available. In the unlikely event that no cache resources can be reclaimed, the cache is bypassed. In this case, CacheFS simply satisfies requests directly from the back file system.
- Since this is a disk based cache, it is nonvolatile. Unlike memory based caching schemes there is no need to repopulate the cache each time a file system is mounted. Likewise, the cache need not be repopulated when the client machine is rebooted.
- Files and directories are populated in the cache as they are referenced. Files are populated in “chunks” rather than being copied to the cache in their entirety. A cached file may be sparse.

CacheFS Performance

CacheFS is not a pure NFS performance accelerator. A single client may not get much of a performance boost from caching, particularly on a lightly loaded fast LAN where the server is a powerful machine with fast disks and not much else to do. The real strength of CacheFS is its ability to reduce network traffic and server load. The benefits of caching really show up on busy networks with loaded servers. In many cases, CacheFS will increase the number of clients that a server can support.

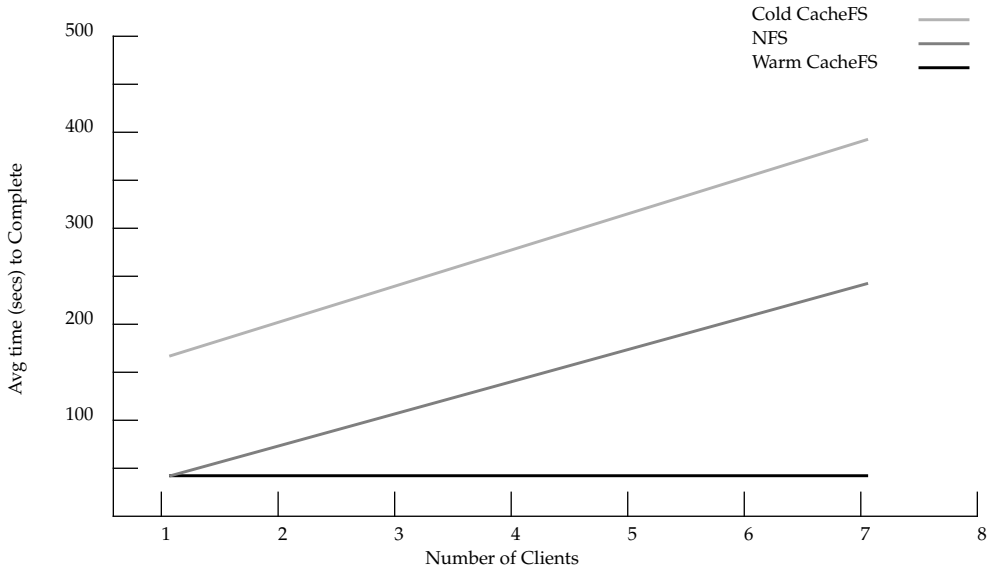


Figure 1 CacheFS Read Performance

The above graph shows the results of a benchmark in which client machines read large quantities of data concurrently from a single server. The benchmark was run once with all clients mounting their data via NFS, once with all clients using CacheFS with a cold (unpopulated) cache, and once with all clients using CacheFS with a warm (populated) cache. As clients are added to the NFS configuration, the average time each client needs to complete the benchmark is increased. When the clients use CacheFS their reliance on the server is drastically reduced such that addition of clients has a minimal impact. A client with a populated cache completes the benchmark in roughly the same time regardless of how many other clients are active.

modifying the automount map files on each client that wants to cache automounts. This allows the question of caching automounts to be decided on a machine by machine basis.

To Cache or not to Cache?

Any file system that is “read mostly” is a likely candidate for caching. Note that file systems where data is generally read one time only (e.g., net news) are *not* good candidates for caching. In this case, you pay the cost of cache population without gaining any of the benefits of subsequent cache accesses.

CacheFS is strictly a “write-through” (as opposed to a “write-back”) cache. Thus, CacheFS provides no benefit for write (including directory modification) operations. CacheFS write performance will never be any better than that of the back file system.

“Read-mostly” file systems like `/usr/openwin`, `/usr/share/man` and `/usr/local` are excellent candidates for caching. Something like `/var/mail` on the other hand is a poor candidate for caching since modifications are frequent and data is typically read once and then discarded.

The *backpath* keyword specifies the place where volume management has mounted the file system. The *special* parameter of the mount command (`/vol/dsk/cdrom_name/s0`) is the volume management device name for the CD. Note that each CD-ROM will have its own distinct "cdrom_name". Again, see `vold(1M)` for details on volume management.

The other interesting aspect of this example is the use of the `noconst` keyword. By default, CacheFS will always perform cache consistency checking. When the `noconst` keyword is specified, consistency checking is disabled. In this mode, no attempt is made to see that the cache remains consistent with the back file system. The cost of consistency checking is non-negligible (especially when the back file system is on a slow device) and in the case of read-only media like CD-ROM it's completely unnecessary. CD-ROM based file systems are excellent candidates for caching with the `noconst` option. Use care when specifying the `noconst` keyword for NFS mounts though. It's not advisable to run in `noconst` mode when files on the back file system could change. Even though one client may mount a file system read-only, it could still be modified by other clients or modified on the server directly.

CacheFS and AutoFS

For CacheFS and AutoFS to work together in Solaris 2.3, it is necessary to install patch 101329-04. Use your normal support channels to acquire the patch.

Enable caching of automounts by specifying the *fstype=cachefs* mount option in your automount map. Note that CacheFS specific mount options (e.g., *backfstype* and *cachedir*) are also specified in the automount map. See `automount(1M)` for details on automount maps.

The following example shows a direct map entry that will cache automounts of the `/usr/dist` file system (using the same cache we've used in prior examples):

```
/usr/dist -ro,nosuid,fstype=cachefs,backfstype=nfs,cachedir=/cache/cache \
distserver:/export/dist
```

You can specify caching in your global (NIS or NIS+) automount maps if and only if all users of the map will have a cache located in the same place (`/cache/cache` in our example). Since this is not too flexible, we recommend

cache as file system B, file system A may reclaim cache resources that had previously been allocated to file system B. In the following example `/usr/local` is cached into the same cache as `/usr/openwin`.

```
root# mkdir /usr/local
root# mount -F cachefs -o backfstype=nfs,cachedir=/cache/cache,\
            distserver:/export/usr/local /usr/local
```

CacheFS does not require changes to the back file system, so many different types of back file systems could potentially be cached. In Solaris 2.3, you may wish to cache HSFS (CD-ROM) as well as NFS. There are performance benefits to caching slow media such as CD-ROM on faster hard drives using CacheFS. Local file systems (UFS) are usually not cached.

CacheFS Mounts in `/etc/vfstab`

The following example shows how to specify our previous example mounts (`/usr/openwin` and `/usr/local`) in the `/etc/vfstab` file. This allows the mounts to occur automatically at system start-up.

```
#device                    device             mount        FS       fsck mount   mount
#to mount                  to fsck            point        type     pass at boot  options
#
/dev/dsk/c0t1d0s0          /dev/rdsk/c0t1d0s0 /cache       ufs      2    yes     -
nfsserver:/usr/openwin     /cache/cache       /usr/openwin cachefs  3    yes   backfstype=nfs,cachedir=/cache/cache
distserver:/export/usr/local /cache/cache     /usr/local   cachefs  3    yes   backfstype=nfs,cachedir=/cache/cache
```

Caching CD-ROMs

In a default Solaris 2.3 configuration, CD-ROM file systems will be automatically mounted by volume management (`vold (1M)`). The *backpath* option of the CacheFS mount command provides the ability to start caching a file system that is already mounted. The following example shows how to start caching a CD-ROM that has already been mounted by volume management:

```
root# mount -F cachefs -o noconst,ro,backfstype=hsfs,\
cachedir=/cache/cache,backpath=/cdrom/cdrom_name/s0 \
/vol/dsk/cdrom_name/s0 /mnt
```

The `cfsadmin` command creates a sub directory (called “cache” in this example) that contains the CacheFS data structures necessary to allow a CacheFS mount.

CacheFS will manage its resources most effectively in cases where the entire front file system is dedicated to caching (as in the above example), or in cases where the non-cache portions of the front file system are static.

Caching NFS

Once a cache is created file systems can be mounted and cached locally. The file system being cached is called the back file system. CacheFS introduces a new file system type (cachefs) to the `mount (1M)` command. Here is an example of a CacheFS mount request:

```
mount -F cachefs -o backfstype=nfs,cachedir=/cache/cache \
          nfsserver:/usr/openwin /usr/openwin
```

Note the CacheFS specific mount options *backfstype* and *cachedir*. The *backfstype* keyword specifies the type of file system that is being cached. The *cachedir* keyword specifies the cache directory that was created with *cfsadmin -c* in the previous example. For details on CacheFS mount options, see the `mount_cachefs (1M)` manual page.

Now `/usr/openwin` can be accessed just like any other mounted file system. As data in `/usr/openwin` is referenced, it will be copied into `/cache/cache`. There is some overhead on initial reference to data in a CacheFS file system (the cost of cache population), but subsequent references to the same data can be satisfied without access to the back file system. A warm cache provides performance close to that of a local file system, without the installation and administration overhead associated with local file systems. The performance benefits of CacheFS can be dramatic for clients on slow links (e.g., PPP) or for clients of heavily loaded servers.

CacheFS allows more than one file system to be cached in the same cache. There is no need to create a separate cache for each CacheFS mount. In typical usage, you only need to run *cfsadmin -c* once to create a single cache for all of your CacheFS mounts. Be aware that cache resources are managed on a per cache (not a per mount) basis. Thus, if you cache file system A in the same

Terminology

- Back File System - the file system that is being cached. Typically this is a NFS or HSFS file system.
- Front File System - the file system that contains the cached data. Typically this is a UFS file system.
- Cold Cache - a cache that does not yet have any data in its front file system. In this case, requested data must be copied from the back file system to the front file system (i.e., the cache must be populated). An attempt to reference data that is not yet cached is referred to as a “cache miss”.
- Warm Cache - a cache that contains the desired data in its front file system. In this case, the cached data can be returned to the user without requiring any action from the back file system. An attempt to reference data that is already cached is referred to as a “cache hit”.

Using CacheFS

CacheFS is bundled with the Solaris 2.3 software release. CacheFS is not enabled by default. Just like NFS, CacheFS needs to be enabled by the `mount (1M)` command.

Creating a Cache

It is important to note that a cache must exist before a CacheFS mount can be performed. No special disk partitioning is required for cache creation. A cache file system may be created in a subdirectory of an existing file system, or you can dedicate an entire file system to caching. Note that you should always create your caches in mounted local file systems. The `cfsadmin (1M)` command creates a cache. The filesystem that contains the cache is called the front file system. (In Solaris 2.3, the only supported front file system type is UFS). In the following example, we create a new file system and create a cache within that filesystem:

```
root# newfs /dev/rdsk/c0t1d0s0
root# mount /dev/dsk/c0t1d0s0 /cache
root# /usr/sbin/cfsadmin -c /cache/cache
```

For an existing file system, ignore the first two lines of the above example.

Cache File System (CacheFS)

Problem

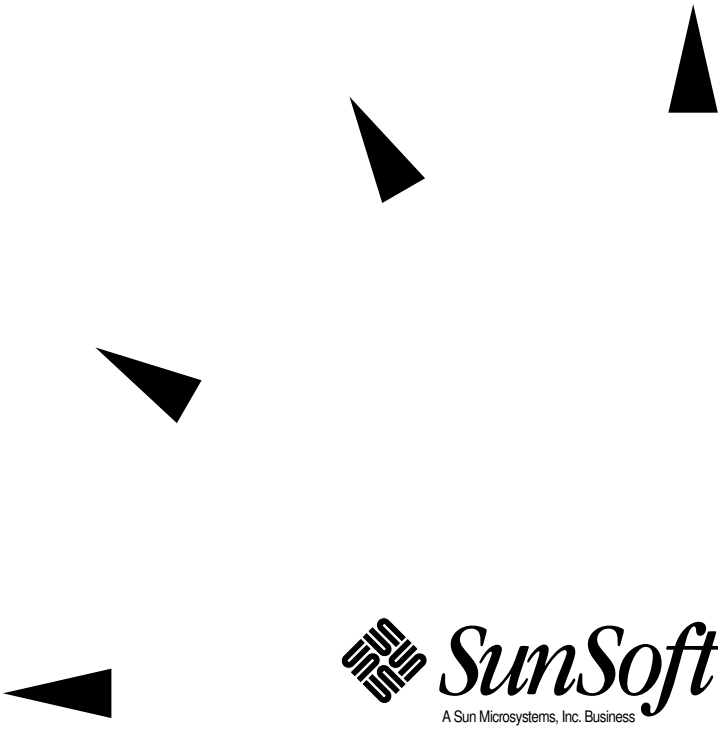
Several factors contribute to overall server performance in an enterprise network, and as the enterprise configuration grows to incorporate more complex capabilities, performance factors play a greater role in the overall effectiveness of the computing environment. A fundamental factor in computer performance is file access time. In networked computer environments, every file access request impacts computer performance. In NFS environments this problem can be mitigated by maintaining a main-memory cache of the most recently accessed data. However, since available physical memory is limited in size, the caching mechanism often has to flush pages and retrieve them again from the server, increasing the load on the server. As more clients are added to the server, the file access time for each client is increased, perpetuating the server performance load, and impacting the overall network performance.

Local disk caching of file systems would reduce the network traffic. Individual client machines would become less reliant on the server, thereby decreasing overall server load. This reduction in server load would lead to an increase in server performance.

Solution

The Cache File System (CacheFS) is a general purpose file system caching mechanism available in Solaris 2.3 that improves NFS server performance and scalability by reducing server and network load. Designed as a layered file system, CacheFS provides the ability to cache one file system on another. In a NFS environment, CacheFS increases the client per server ratio, reduces server and network loads and improves performance for clients on slow links (e.g., PPP). CacheFS also improves performance for file systems on slow media (e.g., CD-ROM).

Cache File System (CacheFS) White Paper



2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Revision A, February 1994

SunSoft
A Sun Microsystems, Inc. Business