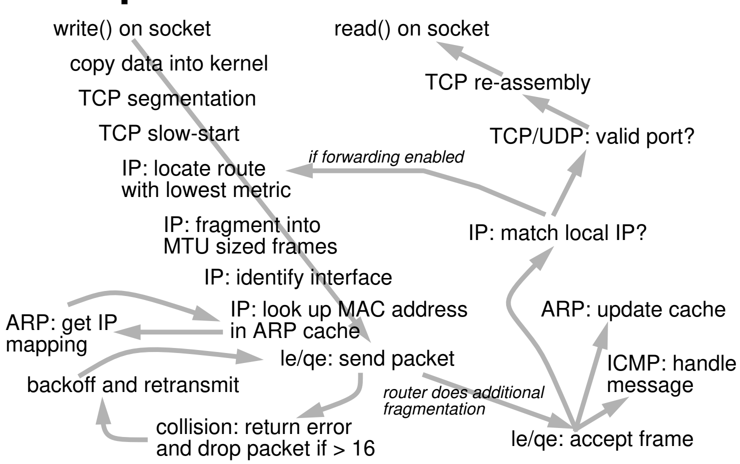
Protocols Exposed: NFS, NIS and Friends

Hal Stern
Distinguished Systems Engineer
Sun Microsystems
15 East Midland Avenue, Paramus, NJ 07652
(201) 967 6111
hal.stern@east.sun.com

Agenda

- Underlying protocols
- Name Services: NIS, NIS+ and DNS integration
- Network File System
- Automounter & CacheFS
- Summary: Capacity & technology planning

Up & Down Protocol Stacks



RPC Mechanics

- RPC uses service (program) numbers and versions
  - Client negotiates highest version of given program
  - Server can have multiple versions, on TCP and UDP
- portmap (rpcbind) maps RPC to TCP/UDP port numbers
  - RPC servers register on startup
  - Clients always find portmap on port 111
  - “getport” requests return registration
  - “callit” requests do indirect call: used for broadcasts

Useful Tools

- etherfind/snoop

    Measures all activity, useful for NIS/NIS+ debugging

```
# snoop -o /tmp/sn to port 111 to dst host sunrise
Using device le0 (promiscuous mode)
# snoop -i /tmp/sn
1    0.00000          kfir -> sunrise        PORTMAP C GETPORT prog=100003
(NFS) vers=2 proto=UDP
2    0.08024          kfir -> sunrise        PORTMAP C GETPORT prog=100005
(MOUNT) vers=1 proto=UDP
```

- rpcinfo -m

    RPC and portmap statistics

- netstat -s

    Show all TCP/IP statistics

More Tools

- rpcspy

    Request-response pairs for any RPC activity

- nfswatch

    Gather request-level NFS activity and sort by host

- AIM products

    SharpShooter: Tuning rules in a simple tool

    AIM III benchmark: synthetic load generation

Name Services: NIS and NIS+

NIS Operation

- Modify getXbyY() calls to use network database

    Replicated on multiple hosts

    Bindings are dynamic but sticky

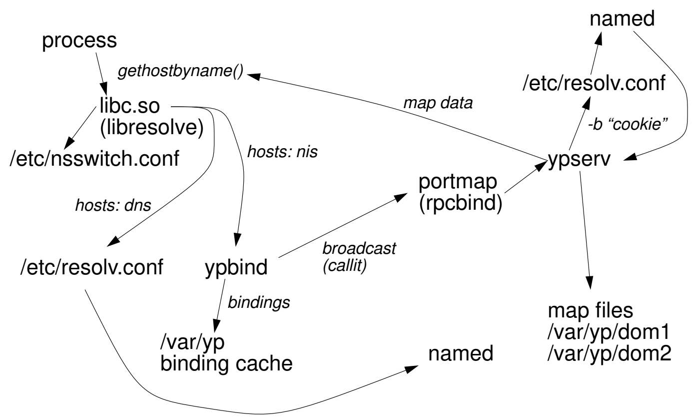
- Name service configuration

    Once NIS is running, it's always used

    DNS integrated on client side or by server

    Solaris: /etc/nsswitch.conf

NIS/DNS Components



Why NIS+?

- NIS is not secure
  - "securenets" features implement basic access control
  - No request authentication on local network
- Match organization and deployment goals
  - Multiple domains, multiple or single points of control
  - Inheritance and "pathing"
- Focus the name service
  - Scalability
  - Reliability - it impacts almost everything

NIS+ Facts

- Robust, secure environment
- By-product is complexity
  - Credential management
  - Server locations, parenting
  - Impact on tools that need naming services
- Name services are becoming more mission critical
  - DCE RPC uses name service (required)
  - Object naming

Managing Complexity

- Principal management
  - Differentiate hosts and users
  - Document mechanisms for changing credentials
  - Keylogin/rlogin/rsh, passwd/chkey exposures
- Change management
  - SCCS or audit trail for ASCII file changes
  - Examine change process for improvement
- Design with redundancy in mind

Domain Hierarchy

- Major source of confusion
  - Each server is member of parent's domain
  - Root servers live in their own domain
- NFS/NIS+ server can have info mismatch
  - Can't authenticate NFS clients on server
- NIS+ client/server mismatches
  - Users lose location-independent permissions
- Define a path/link strategy

Credentials and Authentication

- Think: filesystem model
  - Permissions granted for actions
  - Principals (users) need permissions to operate
  - Directories and objects (files) need to agree
- Credential maintenance
  - Root password changes, server key updates
- Time management is essential for encryption to work
- Inter-domain logins need local authentication

Performance

- Replicas are critical for non-stop performance
  - Must be kept synchronized through checkpoints
  - Position replicas to minimize network traffic
- Name service switching
  - NOTFOUND service search
  - DNS forwarding
  - Links/paths to other domains
- Email dependencies: aliases, inversion, sendmailvars

Transition Strategies

- Run protocols in parallel, in different domains
  - NIS compatibility mode is confusing
  - Name Service Transition kit makes this possible
- Build tools for single/bulk updates of tables
  - nistbladm loads multiple entries into table
  - Automate repetitive tasks
  - Isolate non-recurring ones
- Clearly establish administrative procedures

Multiple NIS Domains

- Multiple NIS domains, on one or more servers
  - Share common maps like passwd, hosts
  - Have per-domain maps maintained on each master
- Centrally manage passwd map
  - Use ypxfr to push master-of-masters copy
  - yppasswd reads master server entry from map

Partitioning Strategies

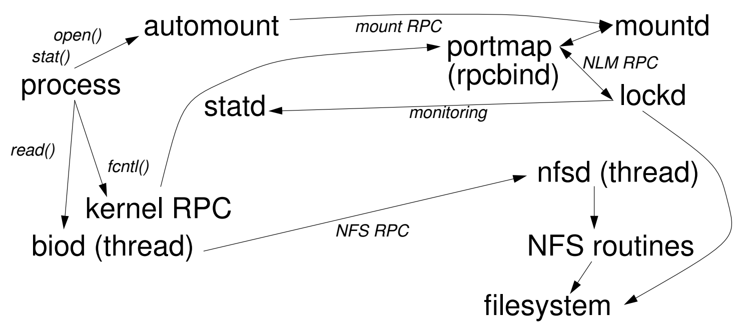
- Configuration versus network management goals?
- Avoid cross-network device bindings
  - In flat network space, put all maps on all servers
- Allow per-domain variations
  - password file
  - automounter maps
- Match names to organizations
  - As seen by system management

NIS+ Status

- Sun is licensing code for ONC+
- X/Open Federated Naming (XFN) to unite services
  - Interfaces constant, like gethostname()
  - Implementations vary over time
- Naming systems continue to evolve

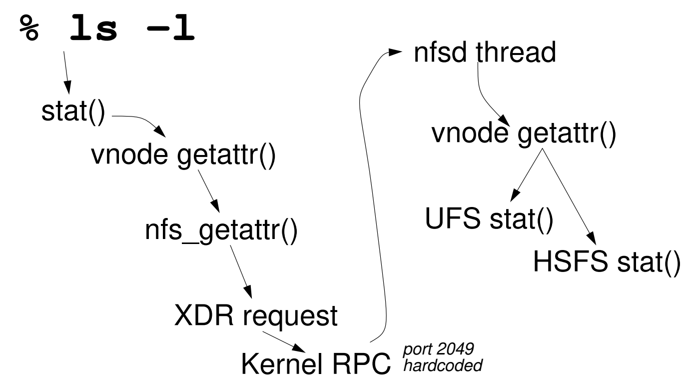
Network File System

NFS Components



- Hard to run NFS through a firewall
- biod/threads cause bursty behavior
- Too few nfsd threads cripples input pipe

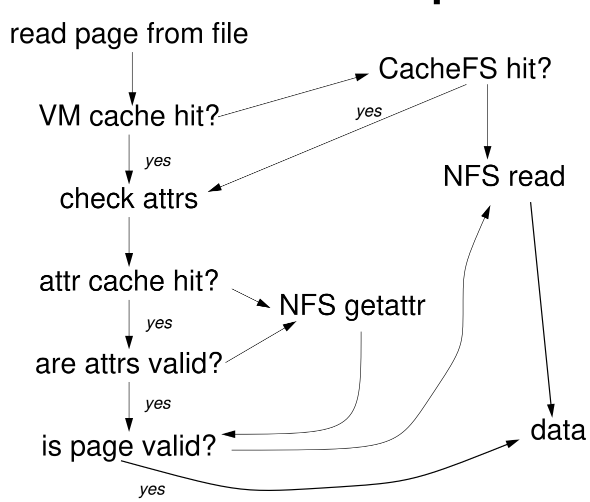
NFS Request Execution



- Kernel RPC uses IP: can get routing errors

  “nfs_server: bad sendreply”

NFS Client Cache Operation



Why NFS V3?

- NFS is a technical “old man” (1985)

  Network bandwidth has changed quite a bit

  Better understanding of network/client behavior
- Filesystems are getting larger

  64-bit file handles are a necessity
- Performance and scalability

NFS V3 Isn't...

- Client-side caching
  - CacheFS in Solaris 2.2 (and later)
  - eNFS (Interstream) in SunOS 4.1.x
- NFS over TCP
  - But V3/TCP is default set of protocol choices
- Hard consistency
  - No new consistency protocol
  - Same rules, but faster
- Unilateral solution: must run on client & server

NFS V3 Features

- 64-bit file handles
- Safe, asynchronous writes for better performance
  - Journal/replay list for recovery
- More atomic operations
  - Exclusive creates, access check (for ACL support)
- Dynamic buffer sizing, up to 32 Kbytes
- Demand reduction
  - Attributes returned in many calls

NFS Over TCP

- Performance gain for large MTU networks
  - FDDI is biggest benefactor
- Doesn't improve peak NFS performance on Ethernet
  - Still gated by Ethernet/WAN speeds for reads
  - Improves write throughput with "streaming" I/O
  - Helps resource utilization
- Makes NFS more tolerant of congestion
  - Better than NFS dynamic retransmission
- Correctness: Many fewer hits in duplicate request cache

NFS/TCP Implementation

- TCP is faster than UDP in Solaris 2.4
  - Single copy, with checksum interleaved
- Inactivity timers
  - 6 minute timer (tunable) on server connection
  - Client reaps connections after low use
  - Next client RPC allocates new socket
- One connection per client-server mount *pair* (tunable)
  - Keeps same port number after reconnect for duplicate request cache operation (port is part of XID key)

NFS Developments

- X/Open Federated Naming (autofs extension)
  - Everything is seen as a file
  - Browse filesystem/name space with NFS
- Removal of authentication dependencies
  - Choose authentication protocol(s) on export
- New lock manager (Solaris 2.5)
  - Status daemon in user space, lockd in kernel
- Access control lists
  - POSIX semantics, extension to V2 and V3

TCP/IP Implementations

- Low overhead, and getting lower
  - Protocol stack was 50% of latency, now < 20%
- Is zero-copy effective?
  - Zero-copy: new VM overhead & MT contention
  - Zero-copy gated by instruction execution speed
  - Single-copy governed by block copy speeds
  - UltraSPARC is 3-4x instruction, 10x block copy
- IPng: Sun's Simple IP+ (SIP) proposal, superset of TUBA
  - Interoperable with old IP, but with 128 bit addresses

Automounter & CacheFS

Why automount?

- Configuration information is complexity
  - "Say it with automount"
  - Standardize and document configurations
- Load distribution
  - Multiple copies of frequently used files
  - Server weightings in Solaris 2.x

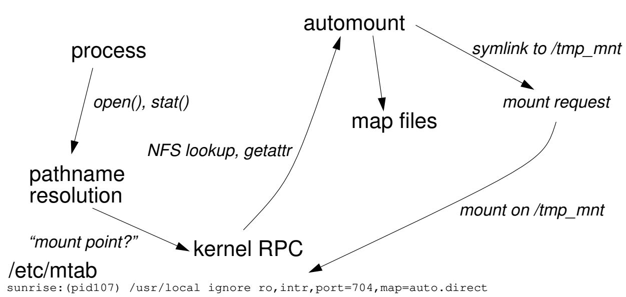
What’s autofs?

- New filesystem type
  - Replaces daemon with an in-kernel filesystem
  - Daemon manages map entries
- Performance and transparency
  - Mounts done in place, not in /tmp_mnt
  - Users can use cd without "falling over edge"
  - Much faster without kernel-daemon interaction
- automountd daemon
  - Manages mount points and mount operations
  - Stateless - easy to update

Other autofs Features

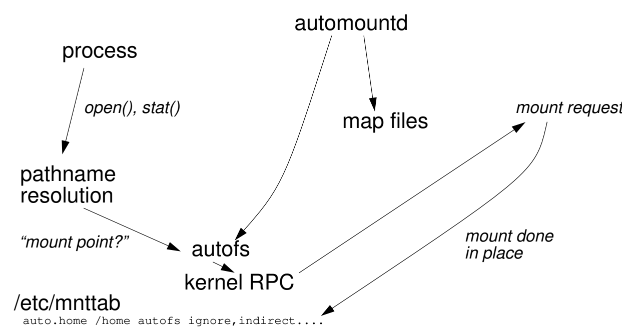
- Filesystem type in map entry
  - automount CD-ROM, cachefs entries
- Executable maps
  - File in /etc with execute bit set
  - Script is run, and map produced on the fly
  - Big win for multiple domains, multi-homed hosts

Automounter Operation



- Automounter only called on first access
- Looks like NFS server with non-standard port number
  - “NFS server pid107@sunrise not responding”

Autofs Operation



Is amd Really Better?

- “Move aside” feature of amd
  - Non-responding server mount point renamed
  - Client mounts alternate server, if replicated
- When does it work?
  - When you don’t need process waiting on NFS server
  - When there’s a replica available

What To Cache?

- Read-only or read-mostly
  - Cachefs is write-through
  - No benefit for regular writes or updates
- Randomly accessed or small data sets
  - Large files fill/flush cache
  - Sequential acces: local disk write without read-back
- Frequently used files
  - Reduce network load
  - Reduce server load

Cachefs “Best Case”

- Local disk access faster than network+server disk
- Network contention affects performance
- Typical working set doesn’t cause paging
  - Contention between paging and cachefs?
  - Performance gated by data fetch, not VM system
- Less benefit/excess cost of additional client system memory
- Varied access patterns: multiple caches
- No consistency checking required (noconst mode)

Autofs and Cachefs

- Replicated servers create replicated cache entries
  - Cold start penalty for switching servers
- What is goal of replication?
  - Load distribution: “corral” bindings with variables
  - Availability: cold cache penalty small compared to down time/service hiatus
  - How many servers will a client find over N days?

CacheFS Performance

- Before/after profile of NFS operations/client
  - Same workload
  - Long time frame (6-8 hours)
  - Conversion of reads into getattrs
- Server load and network latency
- “df” on cache filesystem

Summary: The Big Picture

Capacity Planning

- Separate data access and transfer paradigms
  - Data movement or browsing intense?
  - Many small requests or few large ones?
- Small packets are affected most by large latency
  - Ethernet > FDDI for 100 byte messages
  - Multiple Ethernets provide redundancy, scaling
  - ATM looks promising when LAN emulation works!

Building The Desktop Model

- Similar to sizing a database system
  - Focus on network activity per event, not disk I/Os
  - Determine per user network load
- Translate business transactions into NFS world
  - User transactions and resulting NFS workload
  - Time distribution of transactions
- Weight of each transaction
  - Network traffic measured by snoop/etherfind

Rules Of Thumb

- Ethernet handles about 300 NFSops/second
  - Assuming 30% reads, 15% writes
  - Your mileage will vary
- FDDI does about 1,500 NFSops/second
  - Will improve with NFS V3
  - Wider, not necessarily faster, pipe
- What is the average “pipe size” for a client?

Maximizing Server Disk I/O

- NFS is random I/O intense
  - Distinguish local and global views of requests
  - Software development is sequential I/O in local sense
- Careful administration can balance I/O requests
- Disk striping forces distribution
  - Use 8K byte block size to match NFS request size
  - Is all data accessed uniformly?

Low-Impact Management

- Change control and monitoring
- Performance measurement, monitoring, improvement
- Capacity planning
- Security, reliability
- Developer and user education
- Computing MRP: Where are you going?