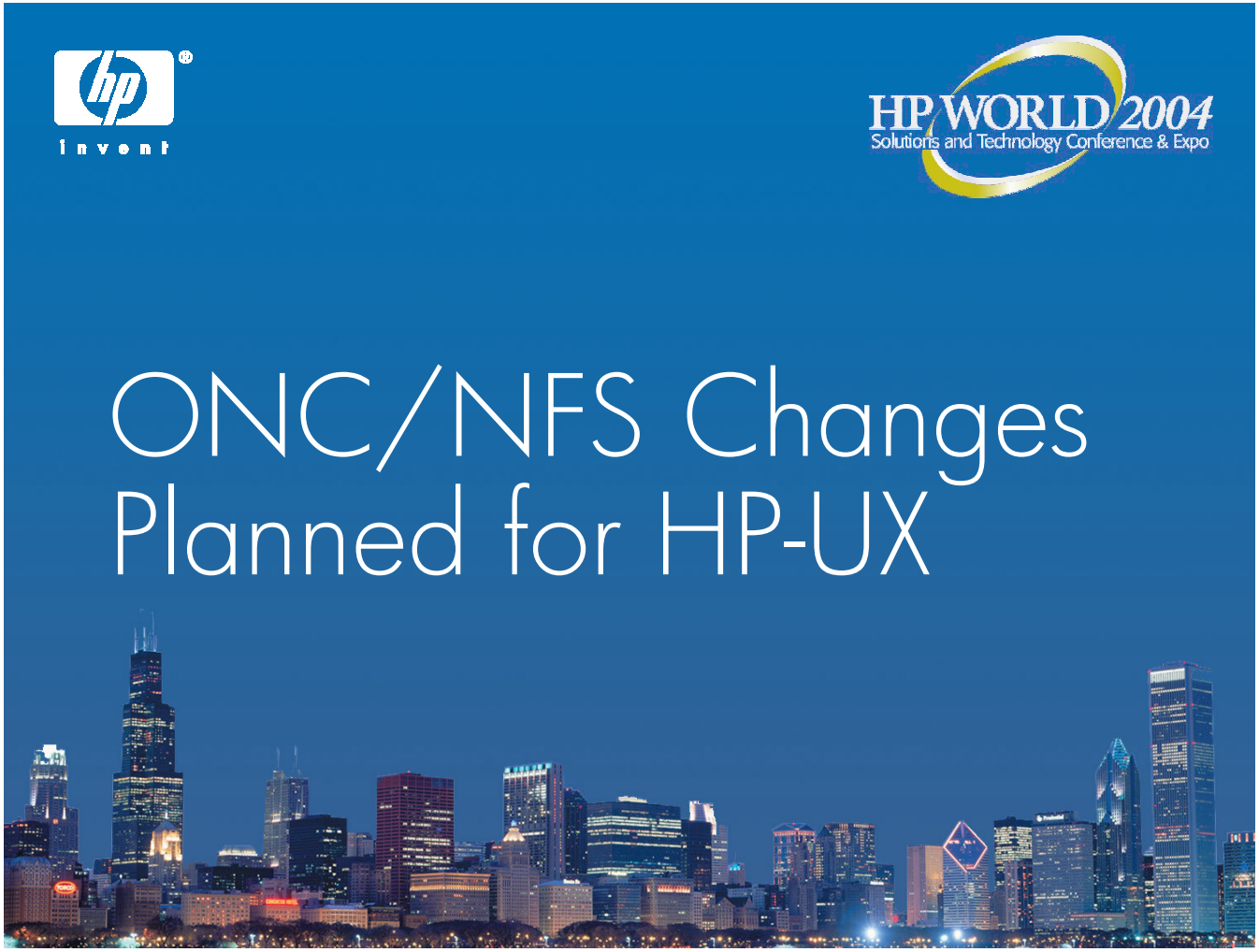

ONC/NFS Changes Planned for HP-UX

Dave Olker
Advanced Technology Center
System Networking and Security Lab

Agenda

- **ONC 2.3 Client-side Enhancements**
  - **NFS Client Kernel Improvements**
  - New Version of AutoFS
  - New Version of CacheFS
- ONC 2.3 Server-side Enhancements
  - NFS Server Kernel Improvements
  - New Version of the Mount Daemon
  - New Version of the Lock Manager
  - New Version of NIS
- ONC 2.3 Features Available in 11i v1/v2

NFS Client Kernel Improvements

- Access Control Lists
- Client-side Failover
- Client I/O Kernel Thread Management
- Local NFS File Locking
- Support for Unified File Cache
- Direct I/O (Non-Cached I/O)
- Attribute Cache Consistency Improvements
- Asynchronous I/O to Locked Files
- Forced NFS Filesystem Unmount

Access Control Lists

An ACL is a list of user ids and group ids with associated read/write/execute permissions on a file for which the ACL is defined. On HP-UX systems, only **POSIX** ACLs are supported.

ACL Behavior on HP-UX 11i v1 and v2

- HP-UX supports ACL's on **VxFS 3.3** (or higher) Filesystems
- The NFS server **will enforce** an ACL placed on a VxFS 3.3 file
- Users on the NFS client **cannot** view or change/set the ACL

ACL Behavior on HP-UX 11i v3

- Users on the NFS client **may** view or change/set a POSIX ACL
- nfsstat(1M) reports the number of **getacl** and **setacl** calls

Client-Side Failover

Allows an NFS client to automatically switch to another server if the original server stops responding because of a hardware/software failure, excessive load, or network fault

Example:

```
mount -F nfs -r bee,wasp:/export/share/man /usr/man
```

- Server switch is **transparent** to users and applications
- Filesystem must be mounted **read-only**
- Filesystems on the servers must be **identical** (use cpio)
- Different from ONC 1.2 AutoFS **"Replicated Servers"** feature (i.e. doesn't require an unmount/remount)

Client I/O Thread Management

- A **separate pool of threads** is allocated to service I/O requests for each mount point
- Threads created/killed **dynamically** based on load
- The number of threads per pool is **configurable**
- A **separate I/O queue** is created per mount point
- Each I/O queue has **sub queues** for different request types (i.e. read, write, readdir, etc.)
- The sub queues are serviced in **round-robin** fashion to avoid starvation of certain request types

Local NFS File Locking

- New NFS mount option will be supported – **llock**
- Instructs the kernel to **not forward** any file locks to the NFS server for any files residing in the specific filesystem mounted with the llock option
- Can **dramatically improve performance** for applications that do lots of file locking
- Only truly safe to use with **read-only** filesystems, since data corruption could occur if multiple clients use local locking and write to a shared file
- Reduces **over-the-wire** RPC calls for file locks

Support for Unified File Cache

ONC 1.2 Split-Cache Memory Architecture

- **Buffer** cache used to store file data
- **Page** cache used to store executables and mmap files
- **Cache coherency** problems with **mmap** files accessed via NFS
- Poor write performance to **mmap** files (sync vs. async)

ONC 2.3 Unified File Cache Memory Architecture

- **Solves** mmap problems of cache coherency
- Makes **asynchronous** writes to mmap files possible
- Improves file **re-write** performance by using smaller block size during read-before-write operations (8K vs. 32K)
- Avoids flushing data caches during **unsuccessful** unmount attempt
- **Simplifies** porting of Solaris ONC code

Direct I/O (Non-Cached I/O)

- **Bypass** Buffer Cache (or UFC) on Client
- Sends WRITE requests using **synchronous** semantics (i.e. FILE_SYNC bit set)
- Most **databases** (i.e. Oracle, Sybase, Informix, etc.) have built-in data caching mechanisms
- **Double buffering** (i.e. once in UFC and once by the application) typically **hurts performance** for most database applications

Attribute Cache Consistency Improvements

- Weak Cache Consistency **Fully** Implemented
- Improved Handling of **Out-of-Order** Attribute Updates
- **Better** Management of Client Attribute and Data Caches
  - If cached attributes are out of sync after a read operation with respect to the server, **only the attribute cache** is purged and not the file cache (or other caches)
  - Better performance by reducing **unnecessary cache purges**
  - Increase NFS **re-write** throughput on MP systems significantly
- **Nanoseconds** Granularity for Cached Time Attributes
- All of the above **reduce** over-the-wire GETATTR calls

Asynchronous I/O to Locked Files

ONC 1.2 Client Behavior Writing to Locked Files

- When an application places a lock on a file, all **data caching** (i.e. buffer cache) and **asynchronous I/O daemons** (i.e. biods) are **disabled** for the locked file
- Required for **data consistency** reasons
- **Dramatically** decreases read and write throughput to locked files

ONC 2.3 Client Behavior Writing to Locked Files

- When an application places a lock on the **entire file** then caching and asynchronous I/O are **enabled**
- When an application places a **byte-range** lock on a portion of the file then caching and asynchronous I/O remain **disabled**

Forced NFS Filesystem Unmount

- New **"-f"** option to the umount(1M) command
- Useful for recovering from "**hung**" or "**stale**" NFS mount points without requiring a client reboot
- The filesystem being forcibly unmounted simply **disappears** from the namespace

Caveats:

- Any existing processes using the filesystem are returned an I/O error (EIO)
- Any locks held by the NFS client for files residing in the forcibly unmounted filesystem are released
- Any data being written to the unmounted filesystem that has not been committed would be lost

Agenda

- **ONC 2.3 Client-side Enhancements**
  - ✓ NFS Client Kernel Improvements
  - ➢ **New Version of AutoFS**
  - – New Version of CacheFS
- ONC 2.3 Server-side Enhancements
  - – NFS Server Kernel Improvements
  - – New Version of the Mount Daemon
  - – New Version of the Lock Manager
  - – New Version of NIS
- ONC 2.3 Features Available in 11i v1/v2

New Version of AutoFS

- On-demand Mounting of Hierarchical Filesystems
- Browsability
- Native Support for Device IDs
- Concurrent Servicing of Mount and Un-mount Threads
- Reliable NFS Ping
- Support for Managing HP CIFS Client Filesystems
- Supports Disabling LOFS Mounts (HA/NFS)
- Support for ONC 2.3 Client-side Failover
- LDAP Support for Map Distribution
- Maintain Support for HP-specific Logging Facility

On-demand Mounting of Hierarchical Filesystems

ONC 1.2 AutoFS Behavior with Hierarchical Maps

- Hierarchical filesystems (i.e. /net –hosts) are mounted in **unison**
- Once mounted, **all members** of a hierarchy must be unmounted **together** or remain mounted, resulting in mount/unmount storms
- Keeping these hierarchies intact requires lots of **overhead** on the client, AutoFS, network, rpc.mountd, and the server

ONC 2.3 AutoFS Behavior with Hierarchical Maps

- Only the **top** filesystem in the hierarchy is mounted
- Other filesystems below the top filesystem are mounted **when needed** (i.e. accessed) and may be unmounted **independently**
- Increases performance by preventing **unnecessary** mounting and unmounting of filesystems that are not being used

Browsability

ONC 1.2 AutoFS Behavior with Indirect Maps

- Listing an indirect mount point (i.e. /home) only displays those subdirectories that are **currently mounted**

ONC 2.3 AutoFS Behavior with Indirect Maps

- Listing an indirect mount point displays all directories that could **potentially** be mounted
- Does not actually mount the filesystems from the remote servers unless they are specifically **referenced**
- **Every** entry in the indirect map is **displayed**, whether it is currently mounted or not

Native Support for Device IDs

At unmount time, the kernel sends an umount request to automountd containing the device ID associated with the filesystem being unmounted

AutoFS Unmount Behavior Prior to HP-UX 11i v2

- automountd searches `/etc/mnttab` for entry with matching device ID
- No match is found because `/etc/mnttab` **does not** track device ID numbers
- automountd calls `stat()` against each filesystem to retrieve its device ID
- **Non-responding** NFS servers cause single-threaded automountd to **block 75 seconds** waiting for `stat()` call to timeout

AutoFS Unmount Behavior at HP-UX 11i v2

- Automountd searches `/etc/mnttab` for entry with matching device ID
- Matching entry is identified **without** calling `stat()`
- **Significantly** improves unmount performance and AutoFS availability

Concurrent Servicing of Mount and Un-mount Threads

ONC 1.2 AutoFS Behavior

- Kernel thread sends mount/unmount requests to automountd
- automountd has **limited** support for multiple threads
- automountd uses a **mutex** to ensure that **only one** automountd thread can access the mount/umount routines at any time

ONC 2.3 AutoFS Behavior

- Kernel spawns **new thread** for every mount and unmount request
  - Prevents AutoFS from hanging if an NFS server is unavailable
  - Single thread may block, but won't block all AutoFS threads
- automountd threads can service mount/unmount **concurrently**
- automountd is now a **fully multi-threaded** daemon

Reliable NFS Ping

AutoFS uses an RPC "ping" routine to verify the availability of the NFS server before initiating a MOUNT or UMOUNT request

ONC 1.2 AutoFS

- **Single UDP** "ping" packet with a hard-coded 15 second timeout
  - "ping" can get **lost** on congested networks or with busy NFS servers
  - Results in **failed** MOUNT or UMOUNT requests
  - **Blocks** AutoFS service until "ping" times out
- No way to be certain if NFS server is really available or down

ONC 2.3 AutoFS

- New **"-retry=n"** option to force multiple server contact attempts
- Only blocks a **single automountd thread** while waiting for NFS server
- Ensures more **reliable** communication with server

HP CIFS Client Filesystem Support

- **HP CIFS Client** is HP's client-side implementation of the Common Internet Filesystem using the SMB protocol
- Allows HP client systems to mount filesystems from **WinNT** servers or other **CIFS/Samba** servers
- ONC 1.2 version of AutoFS **did not support** managing HP CIFS Client filesystems
- ONC 2.3 AutoFS **will support** automatic mounting and unmounting of HP CIFS Client filesystems

Disabling LOFS Filesystems

- AutoFS uses **LOFS** mounts when it detects a "loopback" situation (i.e. the requested filesystem resides on the client system)
- Loopback mount scenarios common with **ServiceGuard HA/NFS**
- LOFS mounts **wreak havoc** with ServiceGuard HA/NFS
  - ServiceGuard **does not** attempt to unmount **LOFS** mounts
  - HA/NFS packages that generated an LOFS mount will **fail to migrate** to adoptive nodes successfully

ONC 2.3 AutoFS – HP-Specific "–L" Option

- Recommended for use on **HA/NFS Servers** running AutoFS
- **Disables** AutoFS' use of **LOFS** filesystems, forcing AutoFS to create **Loopback NFS** mounts
- Allows HA/NFS packages to migrate **successfully**

Support for ONC 2.3 Client-side Failover Mechanism

ONC 2.3 Client Supports Client-side Failover

- Server switch is **transparent** to users and applications
- Filesystem must be mounted **read-only**
- Filesystems on the specified servers must be **identical**
- Different from ONC 1.2 **"Replicated Servers"** feature (i.e. doesn't require an unmount/remount)

ONC 2.3 AutoFS Supports Client-side Failover

LDAP Support for Map Distribution

- AutoFS maps have traditionally been distributed among groups of NFS clients via **NIS or NIS+**
- **LDAP** (Lightweight Directory Access Protocol) is quickly becoming the directory server access protocol recommended by most vendors
- LDAP directories will be **supported** for AutoFS map storage and distribution

HP-specific Debug Logging Facility

- Most RPC-based daemons provide **some** debug logging
- With most daemons, the logging mechanism must be enabled at **daemon start time** via the command-line options
- Not helpful for those problems that occur after a long period of operation; difficult to capture relevant information

HP's SIGUSR2 Debug Logging Toggle Mechanism

- Administrator can **toggle** debug logging on and off without killing the daemon
- If used properly, log file only contains **meaningful** data, which significantly helps simplify troubleshooting efforts
- Support for this toggle will remain in AutoFS 2.3

Agenda

- **ONC 2.3 Client-side Enhancements**
  - ✓ NFS Client Kernel Improvements
  - ✓ New Version of AutoFS
  - ➢ **New Version of CacheFS**
- ONC 2.3 Server-side Enhancements
  - NFS Server Kernel Improvements
  - New Version of the Mount Daemon
  - New Version of the Lock Manager
  - New Version of NIS
- ONC 2.3 Features Available in 11i v1/v2

New Version of CacheFS

- cachefspack(1M) Command
- Support for Large Files & Filesystems
- Improved Cache Consistency Checking
- Maintain Support for “rpages” Mount Option
- New Source Code Base Enables Future Features

cachefspack(1M) Command

- Allows administrator to **pre-load** specific files and directories in the CacheFS cache
- Ensures that specified files will be **present in the cache** whenever possible (i.e. assuming front filesystem resources are available)
- **Improves CacheFS performance** for pre-loaded files and directories
- Affords **greater control** over the cache contents
- Similar to a **manual** version of the HP-specific **"rpages"** mount option

Large Files & Filesystems

- All CacheFS data structures will be **64-bit compliant**
- CacheFS will support the maximum file and filesystem sizes supported by the **underlying front filesystem** in which the cache resides (i.e. VxFS or AdvFS)

Improved Cache Consistency Checking

ONC 1.2 CacheFS Cache Consistency Behavior

- The demandconst feature **is not** fully implemented
  - Consistency check done **every time** cachefsstat issued
  - Consistency check done **every time** a file is opened via CacheFS
- Cache must be **deleted** and rebuilt if consistency checking mount options ("noconst" and "demandconst") are changed

ONC 2.3 CacheFS Cache Consistency Behavior

- The demandconst feature is **fully** implemented
  - Consistency checks done no more than **30 seconds** apart
  - Consistency checks are **not done at file open** time
- Mount options ("noconst" and "demandconst") may be changed **without deleting/rebuilding** the cache

HP-specific **"rpages"** Mount Option

HP's Solution to Binary Caching Dilemma

- Instructs the kernel loader to load **entire** application binaries **contiguously**
- **Automatic** – no further configuration or user intervention required
- Only affects **binaries** – data files are not read in their entirety, only executed binaries are fully populated
- Causes **potentially slower initial** load time, but **substantially faster subsequent** load times

Future Features Being Considered

- **cachefslog** Command
- **cachefswssize** Command
- **cfsfstype** Command
- **Disconnected Mode** (i.e. Server Offline) Operation
- **ACL** Support

Agenda

- ONC 2.3 Client-side Enhancements
  - ✓ NFS Client Kernel Improvements
  - ✓ New Version of AutoFS
  - ✓ New Version of CacheFS
- **ONC 2.3 Server-side Enhancements**
  - ➢ **NFS Server Kernel Improvements**
  - – New Version of the Mount Daemon
  - – New Version of the Lock Manager
  - – New Version of NIS
- ONC 2.3 Features Available in 11i v1/v2

NFS Server Kernel Improvements

- Maintain Support for NFS Version 2 and 3
- Maintain Support for NFS over UDP and TCP
- Server I/O Kernel Thread Management
- Support for ACL's
- NFS Server Logging Facility

Maintain Support for NFS Version 2 and Version 3

- NFS version **2 and 3** are supported with ONC 2.3
- **Backward compatible** with previous HP-UX versions
- Compatible with **3rd party** implementations of NFS
- HP-specific features will remain intact
  - **HA/NFS** (i.e. ServiceGuard) support for both PV2 & PV3
  - HA/NFS support for NFS **File Lock Migration**
  - **Unsafe** PV2 **asynchronous** writing
  - Ability to **disable PV3 READDIRPLUS** on server
  - **Copy avoidance** added to READ path

Maintain Support for NFS over both UDP and TCP Protocols

- **UDP and TCP** protocols supported with ONC 2.3
- **Backward compatible** with previous HP-UX versions
- Compatible with **3rd party** implementations of NFS
- HP-specific features will remain intact
  - **HA/NFS** (i.e. ServiceGuard) support for both UDP & TCP
  - HA/NFS support for NFS **File Lock Migration**
  - **Locking** of nfsd **text and data segments** into memory
  - **High-water** memory allocation per UDP end-point

Server I/O Kernel Thread Management

ONC 1.2 Daemon/Thread Architecture

- UDP requests handled by a **fixed pool** of **single-threaded** nfsds
- TCP requests handled by **separate pools** of nfsktcpd kernel **threads**
  - 10 threads per pool maximum per connection
  - Threads only process requests arriving on their connection
- **Separate** STREAMS modules used for UDP (nfsm) and TCP (rpcmod)

ONC 2.3 Thread Architecture

- Both UDP and TCP requests are processed by a **single system-wide pool** of service threads
- Only a **single parent nfsd** daemon will be launched
- Threads launched and destroyed **dynamically** based on demand
- **Single** STREAMS module used for both UDP and TCP (rpcmod)

Access Control Lists

- VxFS 3.3 (or higher) required for **POSIX** ACL's
- nfsd registers support for **NFS_ACL** RPC program number 100227, versions 2 and 3
- Supports ACL management from NFS **clients** (i.e. setting or viewing ACL attributes)
- ACL's are supported on both NFS **PV2** and **PV3**
- **nfsstat(1M)** enhanced to report ACL requests
- Interoperable with **Solaris** ACL management
  - **May not** interoperate with other ACL implementations (as there is no defined standard for ACL behavior)

NFS Server Logging Facility

- Provides **operational logging** for the NFS server
- Analyzes **RPC operations** processed by the server system
- Useful for identifying which **clients** are using server resources
- Filesystems must be **exported/shared** with logging enabled
- Each record in the log file includes:
  - **Timestamp** of the operation
  - **IP address** (or hostname if it can be resolved) of the client
  - **File or directory name** the operation was performed on
  - **Type** of operation

Example:

```
Sun Sep 21 13:11:00 2003 0 ros87252.rose.hp.com 3579
/home/dolker/testfile b _ read r 0 nfs3-tcp 0 *
```

Agenda

- ONC 2.3 Client-side Enhancements
  - ✓ NFS Client Kernel Improvements
  - ✓ New Version of AutoFS
  - ✓ New Version of CacheFS
- **ONC 2.3 Server-side Enhancements**
  - ✓ NFS Server Kernel Improvements
  - ➢ **New Version of the Mount Daemon**
  - – New Version of the Lock Manager
  - – New Version of NIS
- ONC 2.3 Features Available in 11i v1/v2

New Version of the Mount Daemon

- Multiple Threads of Execution
- Ability to Reject MOUNT Requests from Clients
- Supports Versions 1, 2 and 3 of the MOUNT Protocol
- Supports nfsauth Service
- Maintain Support for HP-specific Debug Logging Facility

Multiple rpc.mountd Threads

- A new thread will be spawned for **each request**
- **Maximum number** of threads can be specified via an rpc.mountd command-line option
- rpc.mountd can now service **multiple** mount, unmount, dump, nfsauth, etc. requests **simultaneously**
- Improves MOUNT **performance** on busy NFS servers
- Significantly decreases the likelihood of an rpc.mountd **outage** due to external factors (i.e. DNS, NIS, etc.)
- While a **single thread of execution may block** due to an unavailable resource (i.e. DNS), rpc.mountd will still be **able to service new requests**

Ability to Reject MOUNT Requests

- The new rpc.mountd will support a **"-r"** option
- Instructs the daemon to **reject** any **new** MOUNT requests from all clients
- Any clients with currently mounted NFS filesystems **are not affected**
- Affords the system administrator **greater control** over the server system's NFS resources

MOUNT Protocol Versions 1, 2, 3

ONC 1.2 rpc.mountd

- Supports MOUNT Protocol Versions **1 and 3, not 2**
- Causes **MOUNT failures** with some NFS clients that won't back-off and use Version 1 when MOUNT Protocol Version 2 is not supported

ONC 2.3 rpc.mountd

- Supports MOUNT Protocol Versions **1, 2, and 3**

Supports the *nfsauth* Service

- ***nfsauth*** is a new service that returns information to the kernel about which **authentication mechanisms** are supported for a specific exported filesystem
- Different from access checks done at filesystem **MOUNT** time – these checks are done at filesystem **ACCESS** time
- The NFS server's kernel checks to see if the exported filesystem was exported with any security flavors
  - If none – no *nfsauth* check is performed
  - If flavors exist – the kernel opens a connection to mountd and makes an *nfsauth* call for the specific client, filesystem, and security flavor
    - If the check is **successful**, the results are **cached**
    - If the check **fails**, the NFS request is **rejected**
- The *nfsauth* cache has a **time-to-live** value of **60 minutes**

HP-specific Debug Logging Facility

- Most RPC-based daemons provide **some** debug logging
- With most daemons, the logging mechanism must be enabled at **daemon start time** via the command-line
- For those problems that occur after some period of operation, collecting meaningful data is difficult if daemons must be **killed and restarted** – or if logging must remain running for **long** periods of time

HP's SIGUSR2 Debug Logging Toggle Mechanism

- This mechanism allows the administrator to toggle debug logging on and off **without killing the daemon**
- If used properly, log file only contains **meaningful data**, thereby easing troubleshooting efforts
- Support for this mechanism **will remain in rpc.mountd**

Agenda

- ONC 2.3 Client-side Enhancements
  - ✓ NFS Client Kernel Improvements
  - ✓ New Version of AutoFS
  - ✓ New Version of CacheFS
- **ONC 2.3 Server-side Enhancements**
  - ✓ NFS Server Kernel Improvements
  - ✓ New Version of the Mount Daemon
  - ➢ **New Version of the Lock Manager**
  - – New Version of NIS
- ONC 2.3 Features Available in 11i v1/v2

New Version of the Lock Manager

- Kernel-based Implementation
- Multiple Threads of Execution
- Supports UDP and TCP Protocols
- Supports Synchronous and Asynchronous Requests
- Improved UDP/TCP Port Semantics
- Client/Server Share Lock Support
- clear_locks(1M) Command

Kernel-based Implementation

ONC 1.2 Lock Manager – Servicing File Lock Request

1. Client's kernel invokes KLM in response to application calling *fcntl()*
2. Client's KLM forwards lock request to local user-space NLM (rpc.lockd)
3. Client's user-space NLM sends request to server's user-space NLM
4. Server's user-space NLM sends request to server's kernel via *fcntl()* call
5. Server's kernel places lock on the server's local file via VOP_LOCKCTL

ONC 2.3 Lock Manager – Servicing File Lock Request

1. Client's kernel invokes KLM in response to application calling *fcntl()*
2. Client's KLM forwards lock request to server's KLM
3. Server's kernel places lock on the server's local file via VOP_LOCKCTL

fcntl() **system call**, user-space/kernel-space **context switches**, and **separate NLM/KLM** overhead for each lock request is **eliminated**

Multiple Threads of Execution

ONC 1.2 Lock Manager Design

- **User-space** daemon
- **Single** thread of execution
- **Entire service** blocks if a resource is unavailable (i.e. DNS)

ONC 2.3 Lock Manager Design

- User-space daemon does initialization then invokes **KLM**
- KRPC (Kernel Remote Procedure Call) layer handles **thread** creation for **each new request** and passes thread to KLM
- **Single thread** may block if a resource is unavailable, but KLM service is available to handle new requests

Supports UDP and TCP Protocols

ONC 1.2 NLM/KLM

- All lock requests were sent/received via **UDP**
- Even **NFS/TCP** filesystems use **UDP** for lock requests

ONC 2.3 KLM

- Lock requests can be sent/received via **UDP** and **TCP**
- KLM requests are sent using the same protocol the **NFS** mount is using (i.e. NFS/UDP = UDP, NFS/TCP = TCP)
- TCP KLM connections persist for **5 minutes**

Supports Both Synchronous and Asynchronous Requests

ONC 1.2 NLM/KLM

- Lock requests from HP-UX 11i v1/2 clients sent **asynchronous**
  - Client sends lock and continues processing while waiting for the reply
  - Server sends reply and continues processing waiting for new requests
- Synchronous not possible because NLM is **single-threaded**
  - One blocked request (i.e. DNS, NIS) would halt **all** lock processing

ONC 2.3 KLM

- HP-UX 11i v3 servers can service either **synchronous or asynchronous** lock requests
- HP-UX 11i v3 clients will use **synchronous** locks
- Synchronous semantics are **preferable** as the server uses the existing connection to send a reply rather than create a new one

Improved UDP/TCP Port Semantics

- KLM always uses "well-known" port **4045**
  - **Eliminates file lock hangs** caused by remote systems caching one port number and rpc.lockd registering another – lock requests are sent to the wrong port
- Client/Server port number information is cached with a **configurable time-to-live** value
  - Default is **5 minutes**
  - Configurable via the **"-t"** command-line option
  - When KLM needs to interact another system whose port cache timer has expired, **updated** port information is retrieved from the remote system's rpcbind(1M) daemon

Client/Server SHARE Lock Support

ONC 1.2 NLM/KLM Share Lock Support

- Server supports share locks from **PC** clients
- Share lock support is performed in **user** space
- **No** integration with share locks from CIFS Client filesystems
- HP-UX 11i v1/2 clients **cannot** send share locks

ONC 2.3 KLM Share Lock Support

- Server supports share locks from **PC or HP-UX** 11i v3 clients
- Share lock support is performed in **kernel** space
- **Integrated** with share locks from HP CIFS Client filesystems
- HP-UX 11i v3 NFS clients **can** issue share locks

clear_locks(1M) Command

- On the rare occasion that an NFS client system crashes and **fails to clear** the locks it was holding, those locks are **unavailable** to other applications
- clear_locks(1M) **forcibly** removes all **file**, **record**, and **share locks** created by the specified hostname
- clear_locks(1M) can be run on an NFS **client** (to clear locks on a remote server) or on a **server** (to clear locks on the local system on a client's behalf)
- **Simulates** a client crash recovery sequence
- clear_locks(1M) can only be run as **root**

Agenda

- ONC 2.3 Client-side Enhancements
  - ✓ NFS Client Kernel Improvements
  - ✓ New Version of AutoFS
  - ✓ New Version of CacheFS
- **ONC 2.3 Server-side Enhancements**
  - ✓ NFS Server Kernel Improvements
  - ✓ New Version of the Mount Daemon
  - ✓ New Version of the Lock Manager
  - ➢ **New Version of NIS**
- ONC 2.3 Features Available in 11i v1/v2

New Version of NIS

- IPV6 Support
- Use of reserved ports
- Shadow Password Support
- DNS forwarding mode
- Multi-homed node information in hosts map
- Use of Transport Independent RPC

Agenda

- ONC 2.3 Client-side Enhancements
  - ✓ NFS Client Kernel Improvements
  - ✓ New Version of AutoFS
  - ✓ New Version of CacheFS
- ONC 2.3 Server-side Enhancements
  - ✓ NFS Server Kernel Improvements
  - ✓ New Version of the Mount Daemon
  - ✓ New Version of the Lock Manager
  - ✓ New Version of NIS
- **ONC 2.3 Features Available in 11i v1/v2**

ONC 2.3 Features Currently Available in 11i v1/v2

- **Asynchronous I/O to Locked Files**
  - Available via ONC patches for 11i v1, ships with 11i v2
- **ONC 2.3 AutoFS**
  - Available at http://software.hp.com for 11i v1, ships with 11i v2
- **Device IDs**
  - Available at http://software.hp.com for 11i v1, ships with 11i v2
- **"rpages" CacheFS Mount Option**
  - Available via ONC patches for 11i v1, ships with 11i v2
- **clear_locks(1M)**
  - Available via ONC patches for 11i v1, ships with 11i v2
- **DNS Forwarding Mode in NIS**
  - Available in HP's ONC 1.2 NIS implementation

HP WORLD 2004

Solutions and Technology Conference & Expo

Co-produced by:

interex

shared knowledge • shared power

encompass

AN HP USER GROUP

RECOMMENDED TRAINING VENUE FOR THE

HP Certified Professional