



Linux NFS: The Future

Chuck Lever, *Network Appliance*
Red Hat Summit 2006

- ▶ **Development**
- ▶ **Testing**
- ▶ **What's yet to be finished in NFSv4**
- ▶ **What's coming in the NFSv4.1 standard**
- ▶ **What's cooking in the lab**

▶ Linux Players

- CITI U-M (sponsored by *)
- NetApp
- Red Hat / SuSE
- Groupe Bull (sponsored by IBM)

▶ Larger community

- OSDL
- Cluster file systems

How You Can Get Involved

- ▶ **Push vendors to make Linux a Tier One platform for your favorite applications**
- ▶ **Invest in the community**
 - **Support a Linux kernel developer**
 - **Donate hardware**
 - **Provide testing resources**
 - **Hire contractors to implement your favorite feature, then donate the result**

- ▶ **OSDL**
 - Client/Server: Automated code analysis
- ▶ **Bull**
 - Client/Server: NFSv4 performance & IPv6
- ▶ **Red Hat**
 - Client/Server: Integration and beta testing platforms in Fedora
- ▶ **SuSE**
 - Client/Server: NFSv4, Kerberos
- ▶ **NetApp**
 - Client: database, failover, Kerberos/LDAP
- ▶ **CITI**
 - Server: newpynfs
 - Client/Server: constant interoperability testing

- ▶ **Performance**
 - Reducing on-the-wire ops
 - Improving I/O concurrency
- ▶ **NLM functionality and stability**
- ▶ **Better TCP transport behavior**
- ▶ **Error recovery**
 - Handling ESTALE

What's yet to be finished in NFSv4

- ▶ **Hardening**
 - Delegation
 - RPCSEC
 - Mount point crossing
 - ACLs

What's yet to be finished in NFSv4

▶ Features

- Global name space
 - Referrals
 - Migration & Replication
- Named Attributes

What's coming in the NFSv4.1 standard

- ▶ **pNFS**
 - Striping data across multiple servers
- ▶ **Sessions**
 - “Exactly once” semantics
- ▶ **Directory delegation**
 - Better lookup and readdir caching

What's cooking in the lab

- ▶ **Client and server performance metrics**
- ▶ **Cachefs**
- ▶ **New transports like RDMA**
- ▶ **IPv6**

What's cooking in the lab (cont)

- ▶ **MP enabled client**
- ▶ **Byte-range delegation**
- ▶ **Cluster file systems**
 - **GFS integration**



Linux NFS Client Performance Metrics

Chuck Lever, Network Appliance, Inc.

Quantifying client performance

- ▶ **Cache efficiency**
 - Client invalidates less often
 - Client does more work with fewer ops
 - Better server and network scalability
- ▶ **Network efficiency**
 - Client moves more data in fewer bytes and with fewer CPU cycles
 - More parallelism
- ▶ **Resource efficiency**
 - CPU
 - Memory

- ▶ **In real time**
- ▶ **With a real workload**
- ▶ **With familiar tools**
- ▶ **Without capturing a network trace**
- ▶ **Without impacting performance**
- ▶ **Granular enough**

New Kernel-User Interface

- ▶ **“cat /proc/self/mountstats”**
- ▶ **Uses familiar paradigm**
- ▶ **Does not compromise private name spaces**
- ▶ **Does not race with mount and unmount operations**

- ▶ **Broken down by mount point**
- ▶ **Data throughput**
 - **Count reads and writes, direct and cached**
- ▶ **Data and metadata cache efficiency**
 - **Cache invalidations**
 - **Opens**
 - **Total GETATTR operations**

- ▶ **Data and attribute cache events:**
 - **INODEREVALIDATE, DENTRYREVALIDATE, DATAINVALIDATE, ATTRINVALIDATE**
- ▶ **VFS calls:**
 - **OPEN, LOOKUP, PERMISSION, UPDATEPAGE, READPAGE, READPAGES, WRITEPAGE, WRITEPAGES, GETDENTS, SETATTR, FLUSH, FSYNC, LOCK, RELEASE**
- ▶ **Miscellaneous events:**
 - **CONGESTIONWAIT, SETATTRTRUNC, EXTENDWRITE, SILLYRENAME, SHORTREAD, SHORTWRITE, DELAY/JUKEBOX**

- ▶ **Bytes read and written via standard system calls:**
 - NORMALREADBYTES, NORMALWRITTENBYTES
- ▶ **Bytes read and written from O_DIRECT file descriptors:**
 - DIRECTREADBYTES, DIRECTWRITTENBYTES,
- ▶ **Bytes read and written on the wire:**
 - SERVERREADBYTES, SERVERWRITTENBYTES
- ▶ **Calls to ->readpages and ->writepages:**
 - READPAGES, WRITEPAGES

NFS Metrics: raw data

```
device manray:/home mounted on /manray with fstype nfs
statvers=1.0
```

```
  opts:   rw, vers=3, rsize=32768, wsize=32768,
acregmin=3, acregmax=60, acdirmin=30, acdirmax=60,
hard, nocto, proto=tcp, timeo=600, retrans=2
```

```
  age:    33104
```

```
  caps:   caps=0x1, wtmult=512, dtsize=4096,
bsize=0, namelen=255
```

```
  sec:    flavor=1
```

```
  events: 9085 3816 953 0 3788 955 12363 892544
0 0 0 3081 6 317 2213 4866 0 3788 0 0 450368 3 0 0 0
```

```
  bytes:  5418033152 3655860224 3691831296
2466820096 3691831296 5811023872 0 457076
```

- ▶ **Broken down by transport**
 - Some mount points share the same transport
- ▶ **Op latency**
 - Network
 - Server
 - Client-side queuing
- ▶ **Op Frequency**
- ▶ **Average size of requests and replies**
- ▶ **Length of backlog queue**

- ▶ **Example: TCP socket**
 - Ephemeral port
 - How many rpcbind operations
 - How many TCP connects
 - How long connects have taken
 - How long transport has been idle
 - How many socket sends
 - How many receives
 - How many unmatchable XIDs have been received
 - Average slot table utilization
 - Average length of the backlog queue

- ▶ **Each RPC operation type shows**
 - The name of the op type
 - How many ops of this type have been requested
 - How many transmissions of this op type have been sent
 - How many timeouts of this op type have occurred
 - How many bytes have been sent for this op type
 - How many bytes have been received for this op type
 - How long ops of this type have waited in queue before being transmitted
 - How long the client waited to receive replies of this op type from the server
 - How long ops of this type take to execute (rpc_init_task to rpc_exit_task)

RPC Metrics: raw data

```
RPC iostats version: 1.0 p/v: 100003/3 (nfs)
  xprt:    tcp 1022 0 2 0 0 511552 511552 0 1290524 24136198
per-op statistics
  NULL:    1 1 0 44 24 1 0 1
  GETATTR: 1803 1803 0 237996 201936 7 318 425
  LOOKUP:  1735 1735 0 259312 294980 7 433 1048
  ACCESS:  461 461 0 62696 55320 0 58 97
  READ:    231962 231962 0 33402528 3721522432 5856442
3514354 9373487
  WRITE:   259811 259811 0 5850515144 41569760 64867376
1147610 66024242
  CREATE:  949 949 0 172080 254332 1 274 335
  REMOVE:  949 949 0 141732 136656 88 2112 2581
  RENAME:  3 3 0 616 780 0 2 2
  REaddirPLUS: 7 7 0 1092 8980 0 2 2
  FSINFO:  1 1 0 128 164 0 0 0
  COMMIT:  13869 13869 0 1997136 2108088 65 2493 3752
```

- ▶ **Specialized**
 - Python scripts
- ▶ **Traditional**
 - nfsstat
 - iostat
 - sar & sadc

Sample Python script: “mountstats”

```
$ mountstats /home
```

```
NFS byte counts:
```

```
applications read 191350434 bytes via read(2)
```

```
applications wrote 202504164 bytes via write(2)
```

```
applications read 8850677760 bytes via O_DIRECT read(2)
```

```
applications wrote 6039011328 bytes via O_DIRECT write(2)
```

```
client read 9629955221 bytes via NFS READ
```

```
client wrote 6448049043 bytes via NFS WRITE
```

```
RPC statistics:
```

```
611741 RPC requests sent, 611514 RPC replies received (61  
XIDs not found)
```

```
average backlog queue length: 77
```

Sample Python script: “mountstats”

GETATTR:

```
3748 ops (0%)    0 retrans (0%)    0 timeouts
avg bytes sent per op: 128    avg bytes received per op: 112
backlog wait: 0.0146    RTT: 0.3842    total execute time: 0.4175
```

ACCESS:

```
142 ops (0%)    0 retrans (0%)    0 timeouts
avg bytes sent per op: 136    avg bytes received per op: 120
backlog wait: 0.0070    RTT: 1.3661    total execute time: 1.3943
```

READ:

```
362559 ops (59%)    1 retrans (0%)    0 timeouts
avg bytes sent per op: 140    avg bytes received per op: 26689
backlog wait: 4.6202    RTT: 1.7300    total execute time: 7.4498
```

WRITE:

```
242766 ops (39%)    226 retrans (0%)    0 timeouts
avg bytes sent per op: 29308    avg bytes received per op: 160
backlog wait: 10.414    RTT: 2.8204    total execute time: 13.366
```

Sample Python script: “iostat-ms”

```
$ iostat-ms --attr /home
```

```
op/s          rpc bklog
  1.47                0.00
```

```
getattr:
```

```
ops/s      Kb/s      Kb/op      retrans      avg RTT (ms)      avg exe (ms)
0.009      0.001      0.109      0 (0.0%)      0.384              0.418
```

```
access:
```

```
ops/s      Kb/s      Kb/op      retrans      avg RTT (ms)      avg exe (ms)
0.000      0.000      0.117      0 (0.0%)      1.357              1.385
```

```
3913 inode revalidations, using cache 4.19% of the time
```

```
1168 open operations (mandatory GETATTR requests)
```

```
5.44% of GETATTRs resulted in data cache invalidations
```

```
$
```

- ▶ **Implement support in nfsstat/iostat/sar/sadc**
- ▶ **Documentation in NFS How-To**
- ▶ **Implement NFS-specific tools for analyzing raw data**
- ▶ **Integrate with SystemTap**