# NFSv4 Performance Across A WAN


Chuck Lever & Thomas Haynes,
*Network Appliance*

Spencer Shepler, *Sun Microsystems*

# Talk Outline

- **Part I: State of the WAN Performance Art**
  - **Our assumptions**
  - **Benchmarks**
- **Part II: Strategies for improving WAN Performance**
  - **Helping all versions of NFS**
  - **What's in NFSv4 today**
  - **The Future**
- **Questions**

# Questions Of The Day

- **Why is NFS on a WAN so slow?**
  - **"I'm a telecommuter. I find NFSv3 performance across the country to suck."**
  - **"Whenever I use NFS from home, all other network activity slows to a crawl."**
  - **"Why can't I quickly access my colleagues' files on a remote file server?"**

# Why Intermediate Caching Isn't Enough

- **Remote shared intermediate cache**
  - **Doesn't help files accessed by only a single client (home directory)**
  - **Doesn't help single remote clients**
  - **Doesn't help read-once workloads**
  - **Still requires GETATTRs and LOOKUPs to maintain close-to-open cache consistency**

# Part I: Our Assumptions

- **WAN performance is slow because:**
  - **The client waits for attribute cache validation**
  - **The client waits for access authorization**
  - **The client has purged its cache and must re-read data or metadata**
  - **The client waits for reads or writes to stream across a slow link**
  - **The client waits for network congestion to clear**

# Part I: Checking Our Assumptions

- **NFS congestion test**
  - Is the NFS client a friendly network neighbor on a slow link?
  - Does it hamper its own performance?
- **Does delegation help?**
  - Application performance comparison
  - Reduction in network traffic?
- **Time wait analysis**
  - Which RPCs cause the most wait time?

# NFS Congestion Test

- **Using the "sync" mount option on Linux appears to reduce network congestion**
  - **"Sync" allows only one read or write on the network at a time**
- **Quantify WAN link congestion due to NFS client**
  - **Can such congestion be reduced or eliminated by careful client design?**
  - **Does client detect congestion it causes?**

# NFS Congestion Test

- **One DSL link with VPN tunnel**
- **Four systems, total:**
  - **Local end: NFS client and ping system**
  - **Remote end: NFS server and ping target host**
- **Two traffic streams:**
  - **NFS writes from client system to server**
  - **Pings from ping system to ping target**
- **Quiescent ping RTT = 75-80ms**
- **TCP window size on uplink: 10KB**

# NFS Congestion Test: wsize=4K

- **Allow one 4K write at a time**
  - **Ping $RTT_{avg}$ = 185ms, $RTT_{max}$ = 384ms**
  - **1:59 elapsed to write 1363656 bytes**
- **Allow two 4K writes at a time**
  - **Ping $RTT_{avg}$ = 396ms, $RTT_{max}$ = 550ms**
  - **1:32 elapsed**
- **Allow three 4K writes at a time**
  - **Ping $RTT_{avg}$ = 671ms, $RTT_{max}$ = 1004ms**
  - **1:32 elapsed**

# NFS Congestion Test: wsize=8K

- **Allow one 8K write at a time**
  - **Ping $RTT_{avg}$ = 315ms, $RTT_{max}$ = 629ms**
  - **1:44 elapsed to write 1363656 bytes**
- **Allow two 8K writes at a time**
  - **Ping $RTT_{avg}$ = 920ms, $RTT_{max}$ = 1105ms**
  - **1:32 elapsed**
- **Allow three 8K writes at a time**
  - **Ping $RTT_{avg}$ = 1479ms, $RTT_{max}$ = 1661ms**
  - **1:32 elapsed**

# NFS Congestion Test: Conclusions

- **Smaller r/wsize is more friendly**
  - **Allows other traffic to interleave**
- **More than two concurrent write requests doesn't improve throughput**
  - **Two requests fill TCP window**
- **Existing congestion control algorithm doesn't help**
  - **Little RTT variance prevents triggering congestion control**

# Time Wait Analysis

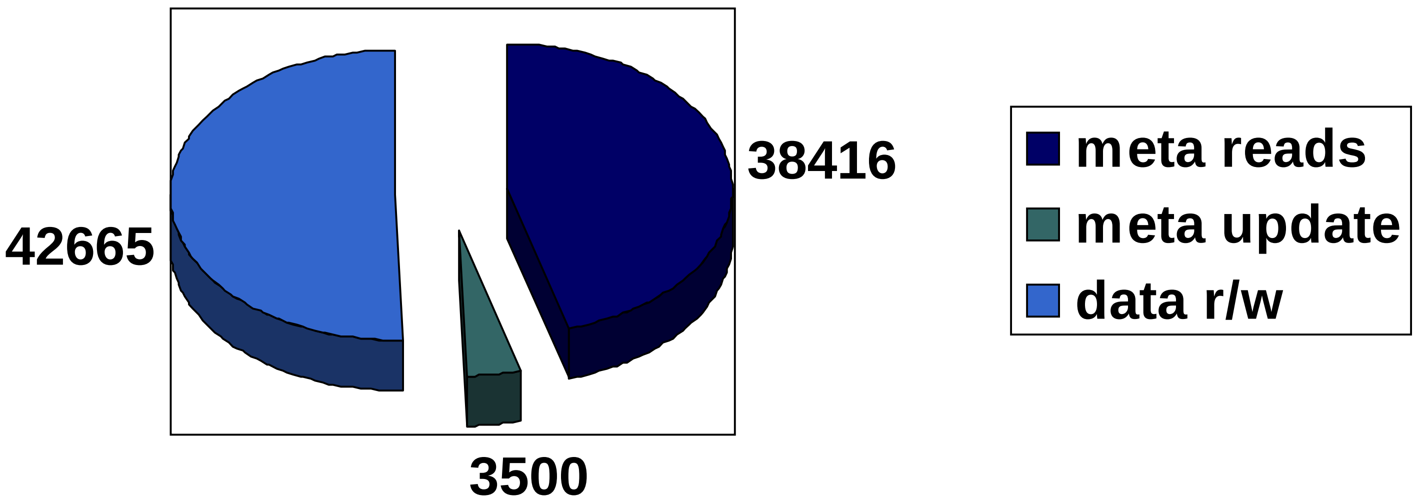
- **Hypothesis:**
  - RTT matters for all workloads
  - Data throughput matters only for data intensive workloads
- **How long does client wait for:**
  - Metadata reads (synchronous)
  - Metadata updates (synchronous)
  - Data reads and writes (sync and async)

# Time Wait Analysis

- **Client: Linux 2.4.20 with readdirplus and access (cold cache)**
- **Server: NetApp F880**
- **Mount options: vers=3,tcp,rsize=4K,wsize=4K**

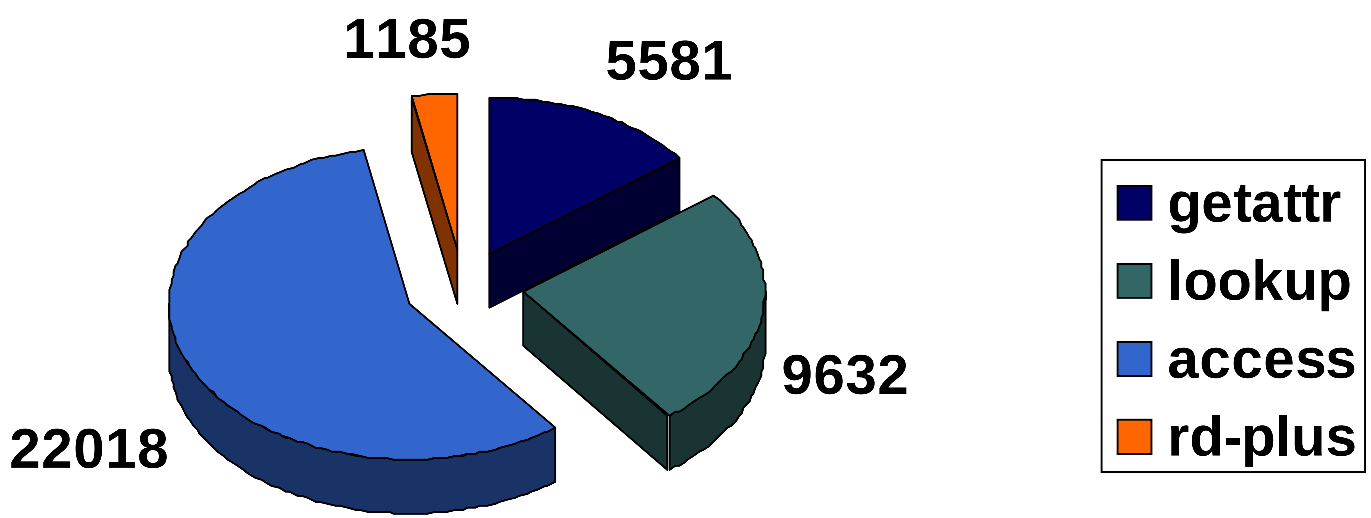
# Time Wait Analysis

## RPC count (total)



# Time Wait Analysis

## RPC count (meta reads)



# Time Wait Analysis

- **84,550 RPCs total**
  - **Almost half are synchronous metadata reads**
- **LAN: 84K RPCs * 45% * 1ms = 38 seconds**
- **WAN: 84K RPCs * 45% * 100ms = 63 minutes**

# Time Wait Analysis: Conclusion

- **Prefetch metadata (VFS cooperation)**
- **More effective cache revalidation**
- **Parallelize synchronous RPCs (access + getattr, access + lookup)**
  - **NFSv3: parallel issue, NFSv4: compound RPC**
- **Depend on "bulk stat"**
  - **NFSv4: compound RPC**
- **Delegation**

# Delegation Test

- Solaris NFSv4 with file delegation
- tar cf -/tmp /usr/include
- ~2100 files, ~150 directories

# Delegation Test

- **Without delegation**
  - **Cold cache: 10.5 seconds elapsed, ~50K ops**
  - **Warm cache: 5.9 seconds elapsed, ~24K ops**
- **With delegation**
  - **Cold cache: 11.2 seconds elapsed, ~50K ops**
  - **Warm cache: 2.5 seconds elapsed, ~7.7K ops**

# Delegation Test: Conclusion

- **After cache is warm, delegation effectively reduces network traffic**
- **Second run with delegation uses 85% fewer RPCs**
- **WAN: (cold) 50K RPCs * 100ms = 84 minutes**
- **WAN: (warm) 7.7K RPCs * 100ms = 13 minutes**

# Part II: Strategies

- **General help for all versions of NFS**
- **Using new features in NFSv4 (RFC 3010)**
- **Driving future versions of NFS**

# Part II: Helping All Versions

- **Eliminate superfluous network requests**
  - **Reduce GETATTRs and LOOKUPs to bare minimum**
  - **Make good use of READDIRPLUS**
  - **Invalidate data and attribute cache less aggressively**
- **Reduce round-trips**
  - **Issue synchronous RPCs in parallel**
  - **Prefetch attribute data**

# Part II: Helping All Versions

- **Use advanced congestion control**
  - **Congestion more broadly defined to include increases in packet latency**
  - **Use TCP window size to control maximum number of concurrent requests**

# Part II: NFSv4 (RFC 3010)

- **File delegation**
  - **Can greatly reduce READ traffic**
  - **Client can trickle write-backs even after a CLOSE**
- **Compound RPC**
  - **More operations per compound means fewer round trips**
  - **Client VFS architecture limits applicability**

# Part II: Future Ideas

- **Directory delegation**
  - **Purpose: reduce or eliminate LOOKUPs and READDIRs**
  - **Server notifies client when directory entries are created or destroyed**

# Part II: Future Ideas

- **Bulk GETATTR**
  - **Attribute “read ahead”**
  - **Grab attributes for a bunch of files at once**
  - **Much like READDIRPLUS, but not necessarily triggered by getdents()**

# Questions