# Linux NFSv4.1 Performance Under a Microscope

Ming Chen,[1] Dean Hildebrand,[2] Geoff Kuenning,[3] Soujanya Shankaranarayana,[1]
mchen@cs.stonybrook.edu, dhildeb@us.ibm.com, geoff@cs.hmc.edu, soshankarana@cs.stonybrook.edu
Vasily Tarasov,[1,2] Arun O. Vasudevan,[1] Erez Zadok,[1] and Ksenia Zakirova[3]
{vass, aolappamanna, ezk}@cs.stonybrook.edu, kzakirova@g.hmc.edu
[1]*Stony Brook University,* [2]*IBM Research—Almaden, and* [3]*Harvey Mudd College*

**Appears as an extended abstract in the proceedings of the 2014 USENIX LISA conference**

## Abstract

NFS is a highly popular method of consolidating file resources. NFSv4.1, the latest version, has improvements in security, maintainability, and performance. We present a detail-oriented benchmarking study of NFSv4.1 to help system administrators understand its performance and take its advantage in production systems.

Our testbed consists of six identical Dell machines. One NFS server connects to five clients via a 1GbE network. We began with a random read workload, where the five clients randomly read a 20GB NFS file exported by the server for 5 minutes. We observed that, in Linux 2.6.32, each client's throughput started around 22MB/s but gradually decreased to around 5MB/s. We found the culprit to be the clients' read-ahead algorithm, which is aggressive and vulnerable to false-positive error. The clients pre-fetched data not needed and wasted around 80% of network bandwidth. The read-ahead algorithm in Linux 3.12.0 is more conservative and the clients achieved consistent 22MB/s throughput.

Switching to sequential read, we observed a winner-loser phenomenon where three (winners) clients' throughput is 28MB/s, while the other (losers) two's throughput is 14MB/s. The winners and losers differ in multiple runs of the same experiments. This is caused by a HashCast effect on our NIC with eight transmit queues. The Linux TCP stack uses hashing when deciding which TCP flow goes to which NIC queue. The clients with collided hash share one queue, and become losers as each queue has the same throughput. We note that multi-queue NIC are popular nowadays, and HashCast affects multi-queue NIC servers hosting concurrent data-intensive TCP streams, such as file-servers, video-servers, etc.

We benchmarked NFS delegation which transfers the control of a file from the server to clients. We found delegation especially helpful for file locking operations which, in addition to incurring multiple NFS messages, invalidate the locked file's entire client-side cache. Our micro-benchmark showed that NFS delegation saved up to 90% of network traffic and significantly boosted performance. Delegations is expected to benefit performance most of the time as "file sharing is rarely concurrent". But it hurts performance if concurrent and conflicting file sharing does happen. We found that, in Linux, a delegation conflict incurs a delay of at least 100ms—more than $500\times$ the RTT of our network!

We found that writing NFS files with the `O_SYNC` flag, which causes more metadata to be written synchronously, has a side effect on the journaling of `ext4`, and can wastes more than 50% of disk write bandwidth. We also noted that the TCP Nagle algorithm, which trades latency for bandwidth by coalescing multiple small packets, may hurt the performance of latency-sensitive NFS workloads. However, the Linux NFS has no mechanism to turn off the algorithm, even though the socket API supports this with the `SO_NODELAY` option.

By showing how unexpected behaviors in memory management, networking, and local file systems cause counterintuitive NFS performance, we call for system administrators' attention to NFSv4.1's intricate interactions with other OS subsystems. For a more flexible NFS, we urge the NFS developers to avoid hard-coded parameters and policies.