Lessons Learned from NFS

Anna Schumaker
NFS Client Maintainer
March 23, 2017

NetApp

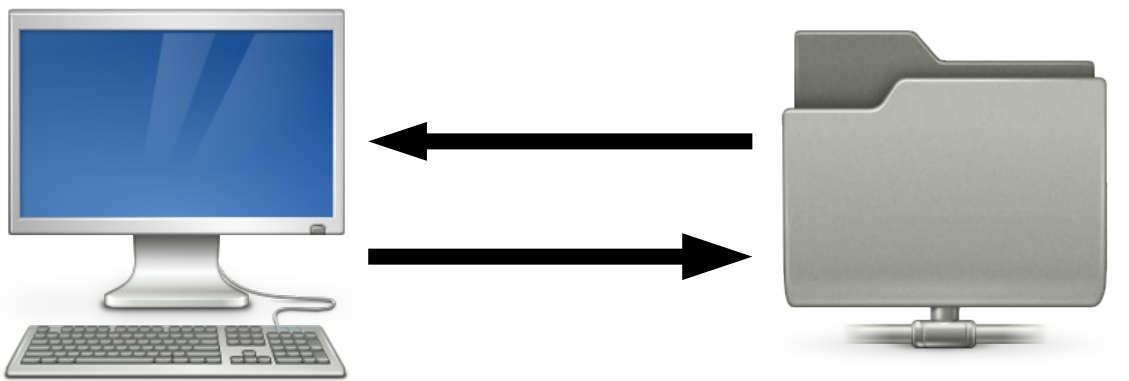
NFS v4.2

- RFC 7862, November 2016
- Several new operations, all optional
- I implemented most of the new operations

Keep the protocol simple

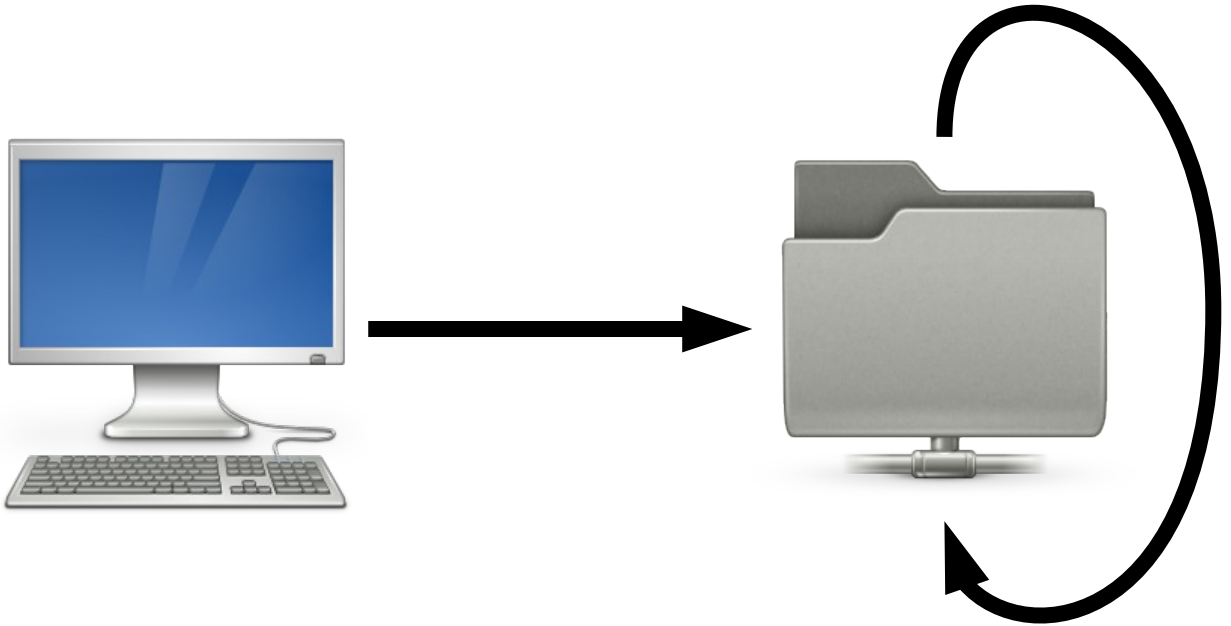
Server Side Copy

- Intention is to speed up copies by avoiding round trips to the client



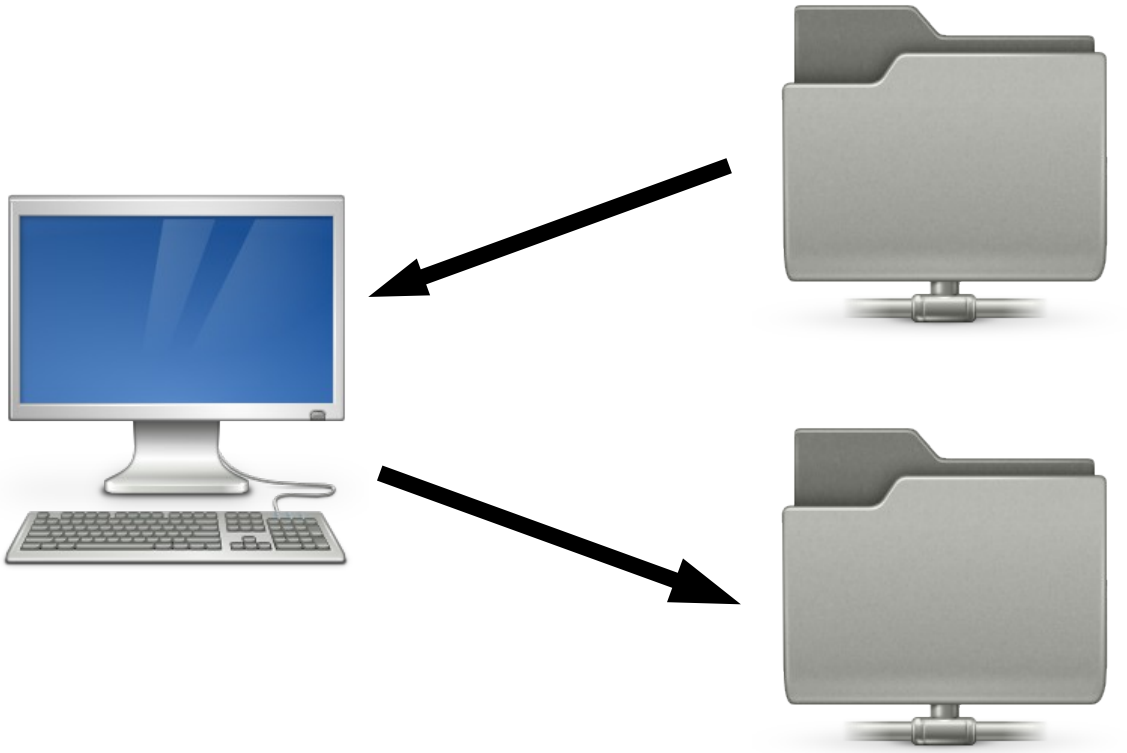
Server Side Copy

- Intention is to speed up copies by avoiding round trips to the client
- Simple case works really well



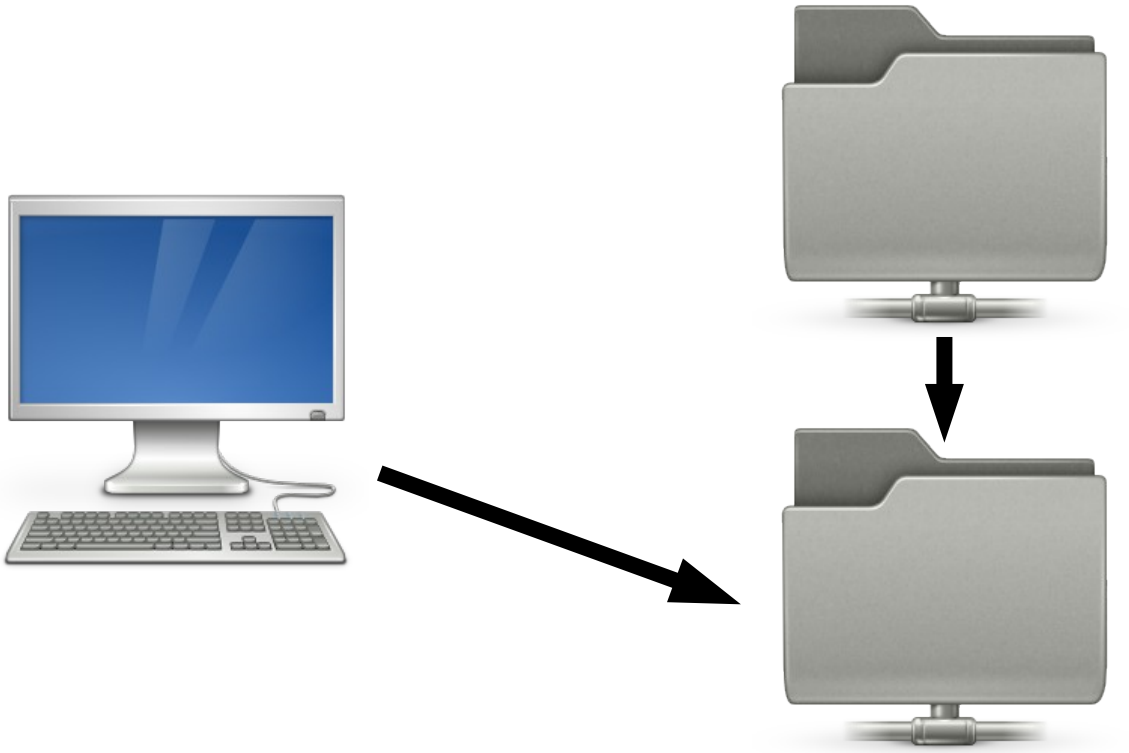
Server To Server Copy

- Intention is to speed up copies by cutting out the client



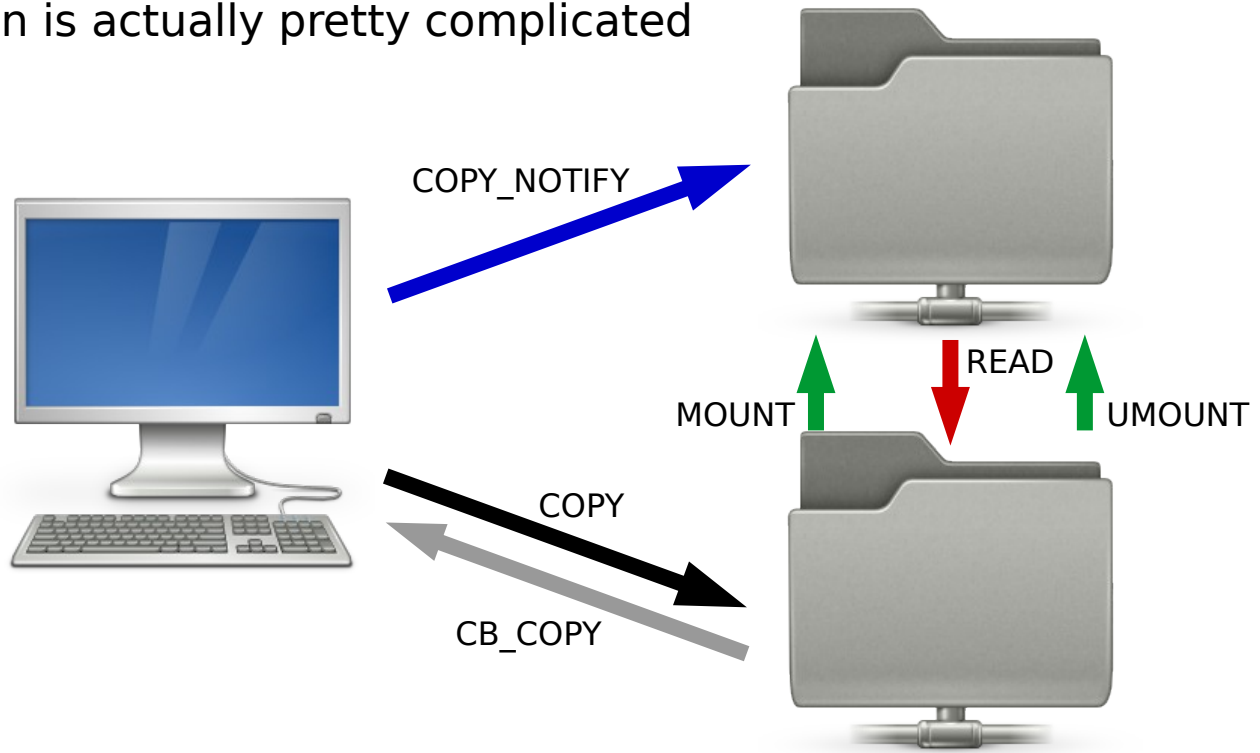
Server To Server Copy

- Intention is to speed up copies by cutting out the client



Server To Server Copy

- Intention is to speed up copies by cutting out the client
- Implementation is actually pretty complicated



NetApp

Keep The Protocol Simple

- Server to server copy is complicated
  - Lots of edge cases and recovery considerations
- RFC is out, but code is still being developed
  - Some issues with the spec are still coming up
- Pushback at allowing the system call to cross mountpoints
- Maybe it would have been better for the spec to only add single-server case?
  - Server to server could have been added later, once spec issues were resolved

Avoid “swiss army knife” operations

Problems with WRITE_PLUS

- WRITE_PLUS was a variant of WRITE
  - Intended for sparse files and Application Data Blocks
- Interface was pretty clunky
  - Each use case needed a different set of arguments
  - Required sparse file operations to be async
  - Reply had unused fields for sparse files

The solution to WRITE_PLUS

- The solution was to split WRITE_PLUS into multiple operations
- Keep WRITE for writing generic data
- ALLOCATE and DEALLOCATE for sparse files
  - Only need offset and length as an argument
  - Returns a status code
  - Mirrors the fallocate interface
- WRITE_SAME for ADBs
  - Still an async operation
  - Still replies with write_response4

Avoid “Swiss Army Knife” Operations

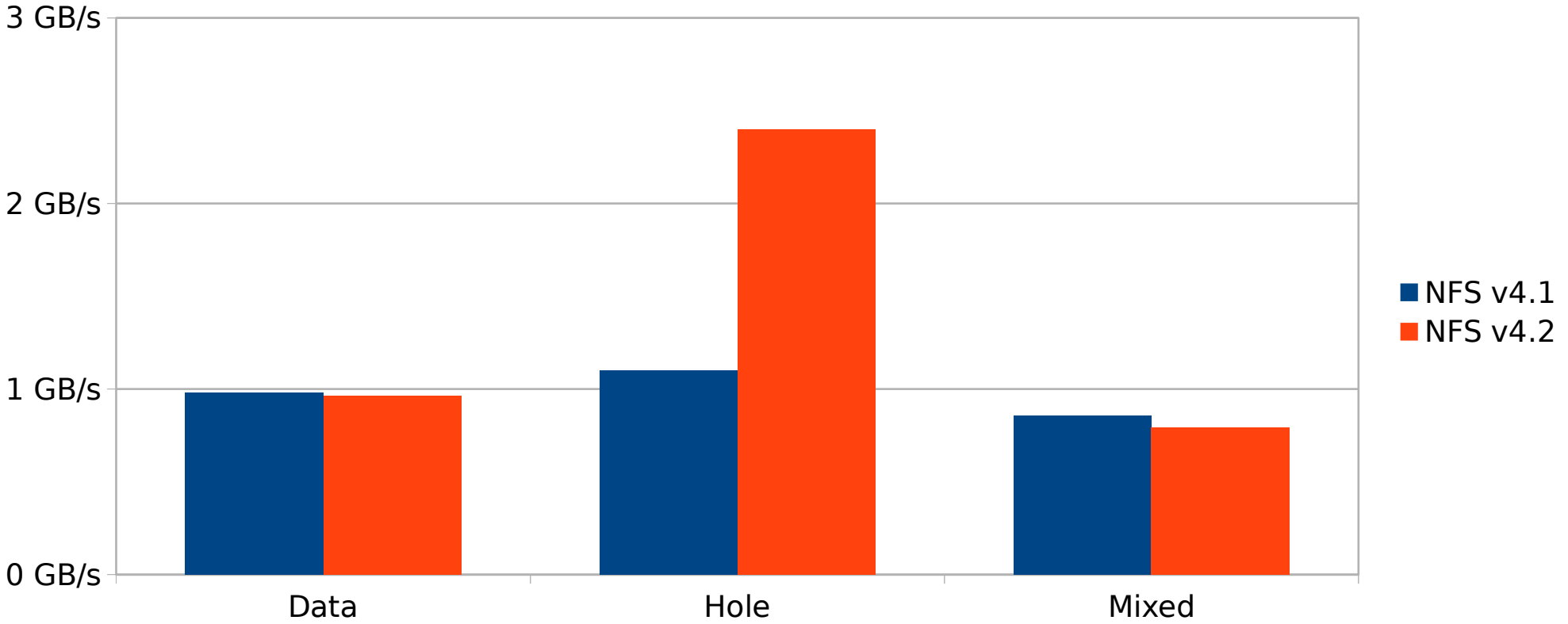
- They make the protocol more complicated than it needs to be
- Group functionality together when it makes sense
  - Don’t put a bunch of unrelated things together

Expect the unexpected

READ_PLUS Performance

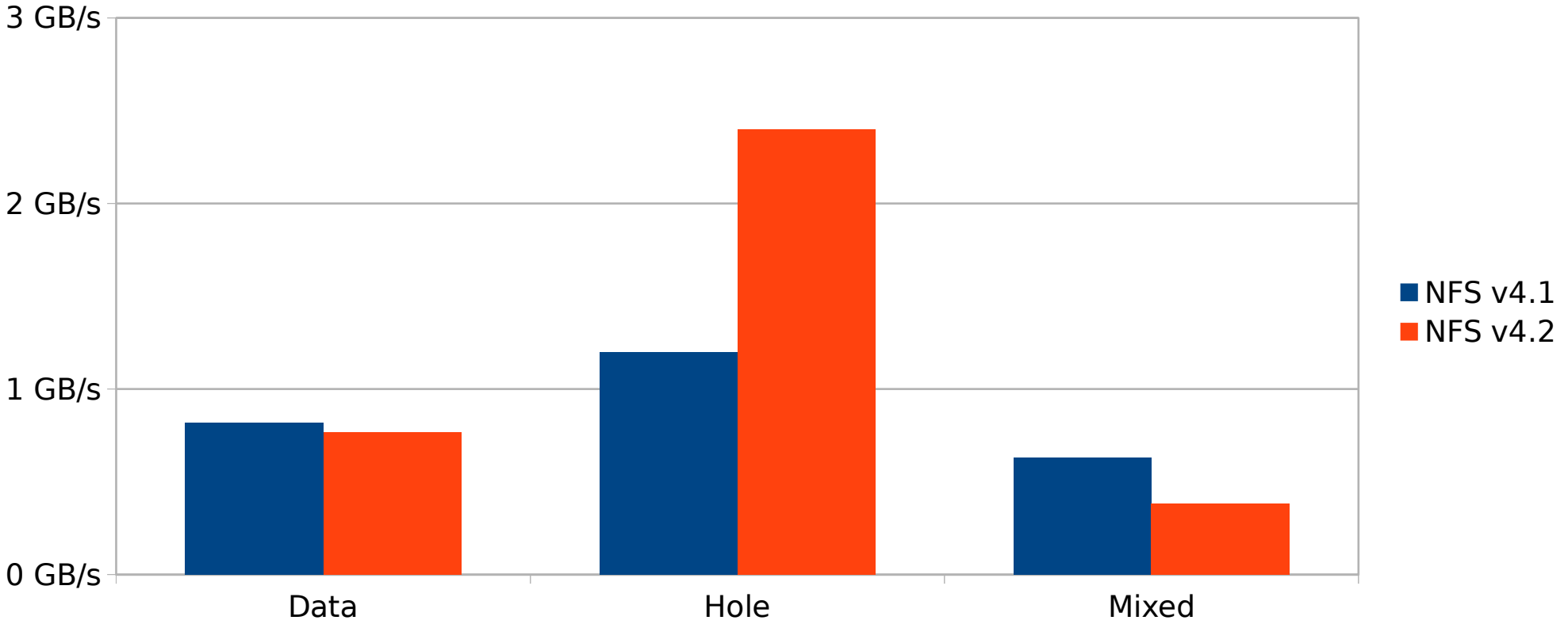
- Intention was to accelerate reading sparse files by compressing out the holes
- Tested three files on EXT4, XFS, and BTRFS
  - 100% Data
  - 100% Hole
  - Alternating data and hole segments
- Works well … if the file is mostly unallocated
  - Not so much when you start mixing data and holes

READ_PLUS + EXT4



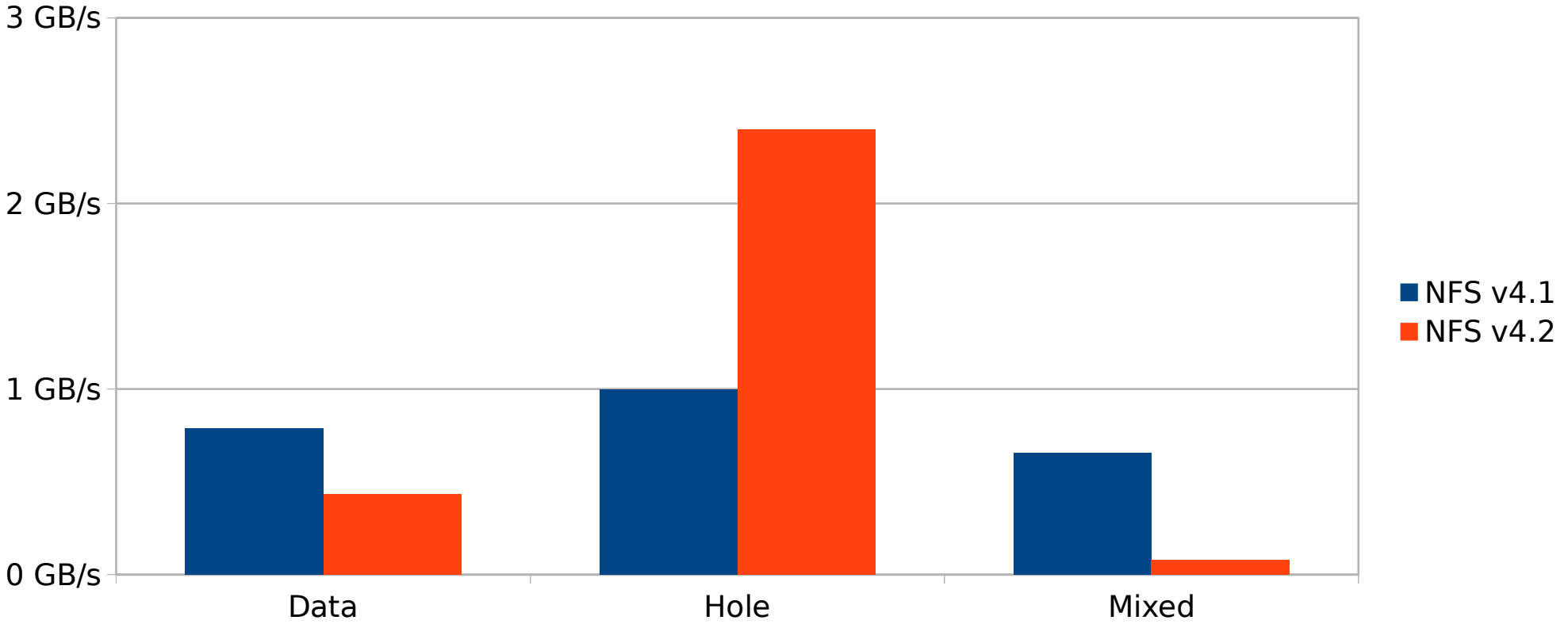
NetApp

READ_PLUS + XFS



NetApp

READ_PLUS + BTRFS



NetApp

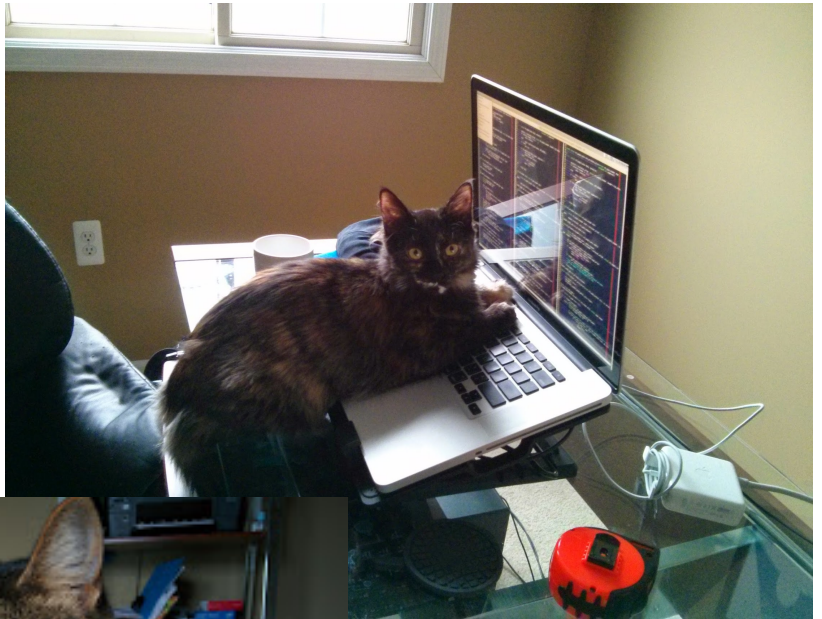
READ_PLUS Problems

- We don’t know the layout of the file in advance
  - Need to seek through it once to find data and hole segments
- Filesystems are taking a long time in lseek
  - Cancels out transferring fewer bytes on the network
- Discussed this at LSF
  - Needs a little more verification on local filesystems
  - Appears to be something that most filesystems can work on
- I update this code against new kernels
  - Won’t be merged until filesystems can keep up

Write code and spec at the same time

Better quality spec

- Most of NFS v4.2 has already been implemented
  - Questions were resolved before releasing the RFC
  - Confusing features were either reworded or removed
- Compare to NFS v4.1
  - RFC released in 2010, but still missing some features
  - Spec questions come up all the time
  - Recently released an updated RFC



NFS v4.2 Extensions

Avoiding NFS v4.3

- Proposal to add new optional operations to NFS v4.2
  - Rather than sitting on them for several years
  - Release RFCs for individual features
  - No other protocol changes
- Servers and clients can support whatever they’re interested in
  - Similar to how v4.2 works now
  - NFS v4.3 can make features mandatory

Extendable NFS Benefits

- Require working code to accept feature proposals
  - Drafts are smaller and more focused
- Demonstrate that real problems are solved
  - Not just theoretical ones
- Develop a better protocol
  - With sensible operations
- Release new features faster