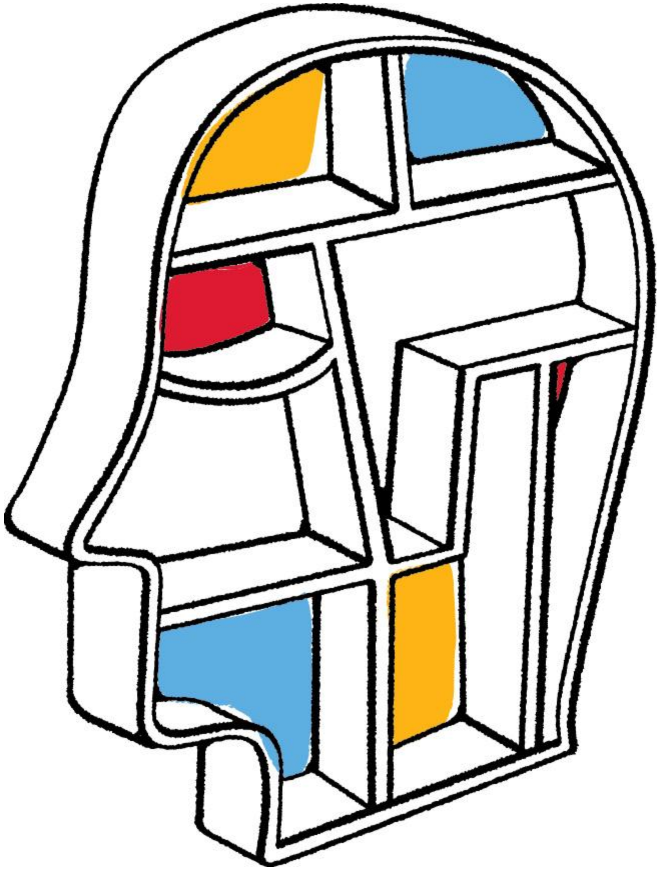
Go further, faster®

Application control of NFS client data caching

Trond Myklebust
<trond@netapp.com>



Why let applications manage kernel caching?

- Applications sometimes have a better idea.
  - Knowledge of the nature of the file
  - Knowledge of the nature of the data
  - Knowledge of when data is being changed

2 use cases

- Clustered applications
  - Distributed lock managers
  - MPI
- Prototyping of new caching mechanisms and policies
  - Emulating new delegation types
    - Writeable directory delegations
    - Byte range delegations

Implementation

- Allows one process at a time to declare itself to be a cache manager for a specific file/directory/symlink
  - Must have write access to the file
- The cache manager then takes over the role of deciding when to revalidate the file caches.
  - Cache manager role is tied to the file descriptor
    - If the file descriptor is closed, the kernel resumes management of the caches

Functionality

- All driven by ioctl()s
- The following functionality is implemented
  - Refresh entire data cache
  - Refresh byte range
  - Flush attribute cache
  - Return delegation (NFSv4/4.1 only)
- ioctl()s apply to files, directories, and symlinks
  - All NFS versions

Future work

- Extend directory cache manager functionality to cover entire subtrees
  - Would allow a single cache manager to act as a lock manager for an entire filesystem or more.
  - Study the effects of writeable directory delegations

NetApp®

Thank you