NFS FS-Cache update

Dave Wysochanski dwysocha@redhat.com

NFS Fall BakeAThon October 2021

Red Hat

Agenda

- Old FS-Cache API, existing problems
- New fscache API (fscache-iter)
- New netfs library / API
- New fscache “fallback” API
- NFS fscache problems

Old FS-Cache API, problems

- Metadata for data being cached
    - fscache does not maintain its own metadata
    - Relies on backing filesystem assumptions about holes and zero extent handling
    - Data corruption is possible, AKA “the bmap() problem”
- IO to backing filesystem
    - Call readpage() on backing filesystem and snoop page waitqueue
- Object state machine is complex
- Page-based API

New FS-Cache API

- Metadata for data being cached
  - fscache maintains its own metadata for data being cached
  - Agnostic to holes / zero block handling of backing filesystem
  - Limitation: cache block granularity set to something like 256KiB
    - Current limitation of 1GB of data/file
- IO to backing filesystem
  - Direct IO via kiocb
- Object state machine greatly simplified
- Request-based API
  - See include/linux/netfs.h

New netfs library / API

- Local caching (fscache)
- Direct I/O, Async I/O
- I/O joining and splitting (see NFS coalesce issue)
- Content encryption and compression
- Invalidation support
- Snapshot & layout support (ceph)
- Takes over VM interfaces
    - eg. AFS's readpage just goes directly to netfs_readpage()
- Hides the existence of pages from the filesystem
    - iov_iter, folios
- The network filesystem provides:
    - Functions to issue data fetch and store requests
    - Functions/info to manage joining and splitting

New netfs library / API

- Recommended way to use new fscache API
- Intended to perform all VM interface services
- A network filesystem (afs, cifs, nfs) defines at least
  - issue_op(): Issue IO to server (i.e. NFS READ)
  - clamp_length(): Set max IO size (i.e. rsize)
    - Inadequate for NFS pageio API
      - NFS may split IO based on various factors
      - Not known at the outset, on page-by-page basis
- A network filesystem's readpage() and readahead() become trivial
- Changes from page-based API (old fscache) to request based API

New fscache “fallback” API

- Temporary alternative to netfs API
    - Temporary ground for existing users of old API (cifs, NFS)
- Page-based, synchronous (performance?)
- Enables removal of old fscache IO
    - A future patchset will remove the old API
- Allows rewrite of existing fscache internals, IO path

NFS fscache problems: netfs API

- NFS cannot tell netfs size of a request when request is created (issue_op)
- netfs calls into filesystem via issue_op()
- filesystem calls netfs via netfs_subreq_terminated()
- Request based, netfs requires we tell it “rsize” (clamp_length)
- The NFS pageio (aka “pgio”) API decides on a page-by-page basis
    - Loop on each page
    - Add page into an existing request?
        - nfs_pageio_do_add_request()
            - nfs_coalesce_size()
                - pg_test()
        - Y: Add page to existing request
        - N: Send RPC with existing pages in request, new request

NFS fscache problems: netfs API

- Why can’t we tell netfs the size of a request (nfs_coalesce_size)?
    - pg_max_retrans (ETIMEDOUT, EIO)
    - incompatible open or lock contexts of nfs_page
    - non-contiguous nfs_page
    - pg_test()
        - pnfs layouts: not known when a request is created
- Possible approaches
    - Terminate a netfs request early (cannot coalesce)
        - Netfs calls issue_op() a second time for remainder of request
            - Existing issue_op() still in progress
                - need to discard somehow, nfs_pageio_do_add_request() fail
    - Additional NFS completion handler logic
        - Call netfs_complete_request() when all NFS requests complete
            - Handle one request success and one fail?
            - Overly complicated?

NFS fscache problems: related

- Convert readpages() to readahead()
  - Assuming NFS patch to use fallback API
  - Just needs patch(es) at this point?
    - readpage() to readahead() conversion
    - NFSIOS_READ* counters
- Handle folios
- Encryption?

Questions!