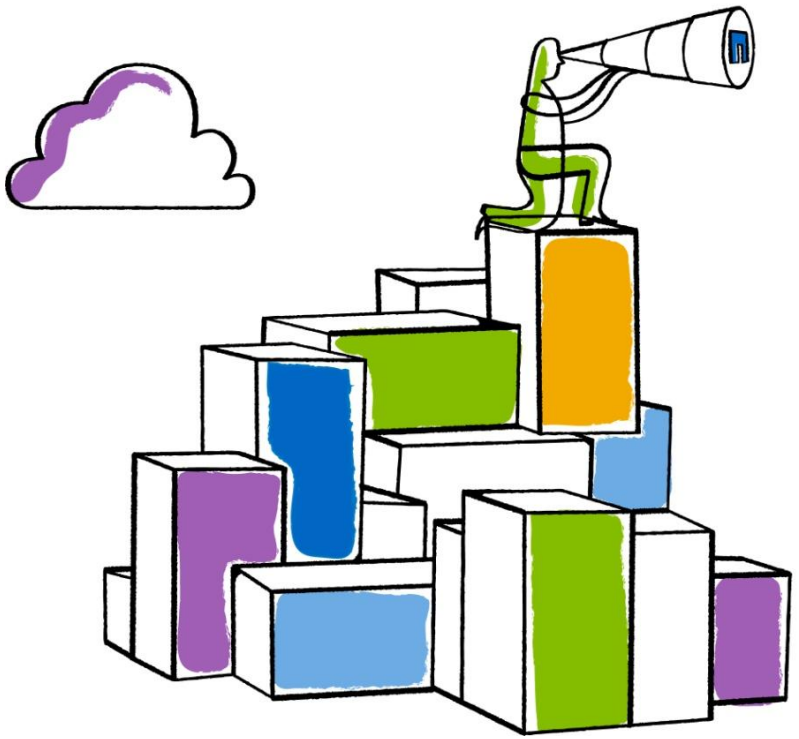
NetApp

Go further, faster®

I/O Hints Discussion



IETF 81

Mike Eisler

mre dash ietf at eisler dot com

July 27, 2011

Overview

- draft-eisler-nfsv4-enterprise-apps-01
- Proposes an IO_ADVISE operation
  - Similar to fadvise()
- Also proposes new READ_WITH_ADVICE and WRITE_WITH_ADVICE operations
- Several controversies

Controversies

- Overlaps with draft-hildebrand-nfsv4-fadvise-02.txt
- Need to provide stronger justification/use case for
  - IO_ADVISE4_PREFETCH_OPPORTUNISTIC
  - IO_ADVISE4_RECENTLY_USED
- The need for READ_WITH_ADVICE and WRITE_WITH_ADVICE

Overlaps with draft-hildebrand-nfsv4-fadvise-02.txt

- Proposed merged list:
  - IO_ADVISE4_NORMAL - as in FADVICE i-d
  - IO_ADVISE4_SEQUENTIAL - as in FADVICE i-d
  - IO_ADVISE4_RANDOM - as in FADVICE i-d
  - IO_ADVISE4_WILLNEED_TO_READ - same as FADVISE_WILLNEED and IO_ADVISE4_PREFETCH.
  - IO_ADVISE4_DONTNEED - as in FADVICE i-d
  - IO_ADVISE4_NOREUSE - as in FADVICE i-d
  - IO_ADVISE4_MIGHTNEED_TO_READ – same as IO_ADVISE4_PREFETCH_OPPORTUNISTIC
  - IO_ADVISE4_WILLNEED_TO_WRITE - same as IO_ADVISE4_INTENT_TO_WRITE
  - IO_ADVISE4_RECENTLY_USED - as in enterprise apps i-d
- To get the IO_ADVISE4_SEQUENTIAL_CACHE behavior, include both IO_ADVISE4_SEQUENTIAL and IO_ADVISE4_WILLNEED_TO_READ in the IO_ADVISE operation.
- To get the IO_ADVISE4_SEQUENTIAL_DONTCACHE behavior, include both IO_ADVISE4_SEQUENTIAL and IO_ADVISE4_NOREUSE in the IO_ADVISE operation.

Justification for IO_ADVISE4_PREFETCH_OPPORTUNISTIC

- Sometimes one is certain a prefetch is needed (e.g. sequential reads), and other times one speculates it is needed
- IO_ADVISE4_PREFETCH is for the certain case
- IO_ADVISE4_PREFETCH_OPPORTUNISTIC is for the speculative case where it costs the server little to perform
  - E.g. an application reads data that contains a reference to data in another block (possibly in another file, possibly in another server)
- A server that is lean on free/cold cache space might prefetch block pointers instead of the block itself

Justification for IO_ADVISE4_RECENTLY_USED

- Data can go cold in the server's cache while it stays warm in the client's cache
- In order to meet service level objectives including in the face of client restart, the server needs to know which data is warm
- Data that gets LRUed out of server's primary cache (e.g. DRAM) can placed in seconday cache (e.g. flash memory)

The need for READ_WITH_ADVICE and WRITE_WITH_ADVICE

- The objective was to handle the case where the client is indicating advice that applies to just one I/O operation and leaves the IO_ADVISE hint intact
- E.g. Overall the file has a random workload, but the client knows when it reads a particular block that the block will be immediately written (e.g. database record update)
  - So server need not cache the block
  - And if the server's file system is log based, this provides advance notice to find free space
- This class of use cases can be handled by doing (for example)

  IO_ADVISE IO_ADVISE4_WILLNEED_TO_WRITE ; READ ; IO_ADVISE previous_hint
  - But this leads to some other issues …

New Issues

- How many hints does a server support per open-owner/file pair
  - Very relevant to whether proposal drops READ/WRITE_WITH_ADVICE
  - Client should know how many

New Issues (continued)

- Proposal 1: IO_ADVISE response to include count of number of hints the server has on the file
  - E.g., a client requests two hints on two non-overlapping byte ranges
    - The second IO_ADVISE response indicates just one hint (the last hint) is in effect
    - If there a maximum of one hint, then since NORMAL is also a hint, then this means that despite the byte range, the hint always applies to the entire file
  - If the server supports between 2 and 2^64 hints then the specification needs to define which of the remaining hints apply to orphaned byte ranges
    - Hint with nearest offset?
    - Least recently sent hint?
    - Most recently sent hint?
    - …

New Issues (continued)

- Proposal 2: Drop byte range from IO_ADVISE arguments
  - Unambiguously dictates that the protocol supports exactly one hint per open-owner/open file pair
  - Much simpler, if limiting
    - LAYOUTCOMMIT provides a lesson here
  - Clashes with POSIX standard for fadvise, but does anyone implement multiple byte ranges?

New Issues (continued)

- pNFS issue: should IO_ADVISE be allowed on requests to data server (DSes)
  - Very relevant to whether proposal drops READ/WRITE_WITH_ADVICE
  - In order to satisfy per I/O hint use case, protocol must allow this possibility
  - Then what does it mean to send IO_ADVISE to both MDS and DS?
  - This is analogous to COMMIT: MDS decides whether pNFS client can COMMIT to DS.
  - Proposal is to solve it the same way
  - New flag NFL4_UFLG_IO_ADVISE_THRU_MDS in the field nfl_util of the file layout
    - if set, IO_ADVISE MUST NOT be sent to DS
    - if not set,
      - IO_ADVISE MAY be sent to DS but will not impact other DSes
      - Hint will not outlive the layout

Proposal for Moving forward

- Since READ/WRITE_WITH_ADVISE are contentious, drop those operations and address multiple hint and pNFS issues
- Combine the two I-Ds (just the hint stuff from the enterprise apps I-D), using merged hint list presented earlier
- New operation is called IO_ADVICE since it supports both POSIX fadvise and non-fadvise requirements
- Incorporate into NFSv4.2

NetApp

Thank you

NetApp

Q&A