Client-Side Direct I/O for NFS

Mike Kupfer

kupfer@Eng.Sun.COM

28 February 1997

Disclaimer

This is not a product announcement.

Overview

- Background
- Changes
- Performance Results
- Future Work, Issues

Background

The Benchmark

what

- sequential I/O: `mkfile` a 60 MB file, then `dd` it to `/dev/null`
- unmounts on client and server to flush caches

why

- LADDIS (SPEC SFS) doesn't measure client
- LADDIS measures aggregate, not point-to-point
  - expect 6+ MB/s on SS10/20 with FastEthernet, only getting 5 MB/s (up from 3.4 MB/s)

Direct I/O

- bypass page cache
  - best for large files, no locality of reference
  - avoid page cache overhead
  - avoid polluting page cache
- UFS Direct I/O project in 2.6
  - databases, decision support software
  - might help NFS server; what about client?

Direct I/O (cont'd)

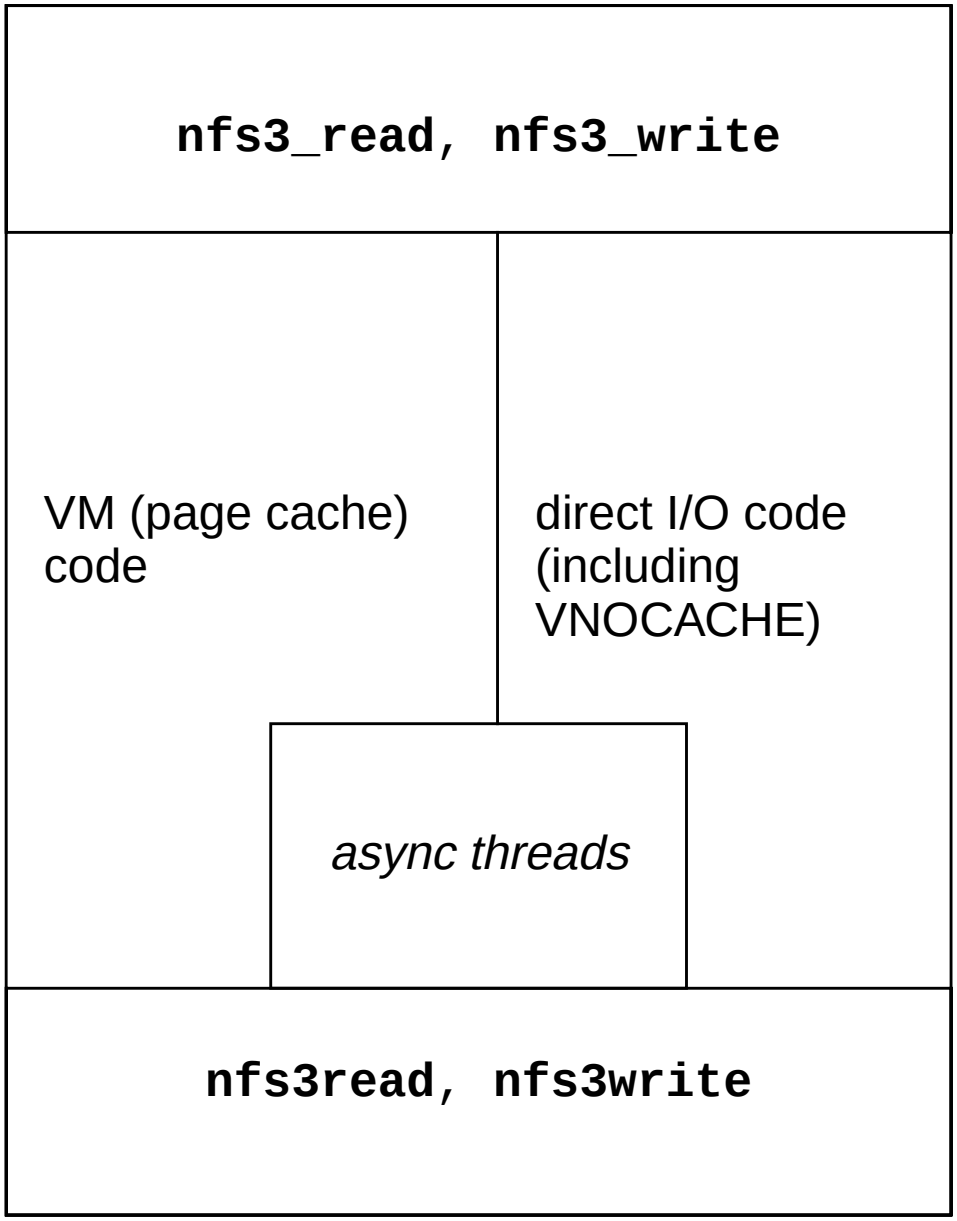
- SGI's Bulk Data Service
  - `O_DIRECT` flag combined with NFS file
  - stuff bytes into a TCP socket connection
  - 60 MB/s over HIPPI (March 1996)
  - uses private protocol, requires client and server changes

Changes

Overview of Changes

- API support: make look like UFS
- add array of buffers to rnode
  - `kmem_alloc`, `kmem_free` buffers as needed
- use buffers instead of VM segment
- keep the pipe full
  - use readahead and write-behind
  - large transfer sizes
  - safe asynchronous writes
- transparent to server except for larger transfer size

Client Structure



Performance Results

Issues, Future Work

Issues

- to productize or not to productize
  - verify on UltraSPARC, other benchmarks
- API for determining transfer size
- tuning
  - how many buffers
  - when to issue `COMMIT`
- MT support too hairy?
  - less arcane scheme for iterating over buffers

Things To Do

- failover support
- cache management, error handling
  - make direct I/O consistent with VM-based code (such as it is)
- misc. cleanup
  - API for enabling/disabling direct I/O
  - code organization
  - plug into kmem reclaim logic
  - coexistence with mmap
  - etc.

Futures

- application-directed readahead?
- page flipping?
- server-side direct I/O
  - assume client cache takes most hits for NFS
  - use UFS direct I/O

Conclusions

- bypassing page cache is a win for sequential access, no locality of reference
- the win gets bigger if the file doesn't fit in memory
- keeping the pipe full is more work, but necessary