A new batch system, dCache and nfs

A. Pickford

Background

- Nikhef Local Batch System (Stoomboot)
  - originally 90 worker nodes
    - Dell M600 blades, 8 cores, 1Gb/s nic, slc6
- dcache system
  - 8 storage systems (820 TB total)
  - started with version dcache 2.13 in 2016, upgraded to 3.1 in 2017
  - nfs v4.1 mounts to dcache on all batch nodes
  - lots of initial nfs issues when stress tested
    - issues fixed and very reliable performance from 2016 to end 2018.
    - see my 2016 workshop talk for some of the details

Nikhef

New Nodes

- Nov 2018: added 25 new worker nodes
  - Dell 6415, AMD EPYC 7551P (32 cores), 256 GB Ram, 25 Gb/s nic, centos 7
  - tested with dcache system – no initial issues
    - BUT not extensively stress tested
- Feb 2019
  - slc6 nodes mostly retired
  - most users starting to work with centos 7 nodes
    - new types of jobs
  - nfs lock ups on new worker nodes
    - multiple jobs on same node opening multiple files in dcache
    - leading to the whole nfs system on client locking up

Nikhef

Belated Stress Testing

- Tested on 8 new worker nodes
  - 24 simultaneous dcache read/writes per node
  - lots of errors on the nfs door

```
24 Feb 2019 13:59:22 (NFS-hooikoorts) [] Bad Stateid: op: LAYOUTRETURN : NFS4ERR_BAD_STATEID :
State not known to the client: [5c701c820000017f00002838, seq: 2]
24 Feb 2019 19:30:18 (NFS-hooikoorts) [] NFS server fault: op: WRITE : NFS4ERR_IO : Mover finished,
EIO
25 Feb 2019 16:09:19 (NFS-hooikoorts) [] Bad Stateid: op: READ : NFS4ERR_BAD_STATEID : State not
known to the client: [5bdb258e0000002d00076c2c, seq: 2]
```

  - and on the pools

```
26 Feb 2019 21:11:17 (kip-05Pool05) [] Failed to send RPC to /2a07:8500:120:e070:0:0:0:3e7:934 :
Connection reset by peer
```

  - on clients
    - nfs kernel threads going into uninterruptible sleep waiting for nfs4_proc_layoutget calls to return

Nikhef

Workarounds

- Tested two workarounds
  - downgrading new nodes to slc6
    - worked: no issues seen on downgraded nodes
    - offered a reduced batch service with some new machines using slc6
    - not a long term solution
  - downgrading to centos 7.3
    - centos 7.4 introduced support for flexfile nfs v4 layout files
    - tested if changes to nfs kernel modules to support flexfile caused problems
    - nfs kernel still locked up with multiple nfs access to dcache on the same client node

Nikhef

dCache 5.0

- Reran stress tests using dcache 5.0
  - already had a dcache 5.0 test system available (planned dpm to dcache migration)
  - tried with nfs 4_1 and with flexfile layout files
    - nfs 4_1 layout files showed similar issues
      - nfs kernel threads still hanging during layout get calls
      - centos 7.4 and later clients also used nfs v3 read/write rpcs to access files
    - flexfile layout (as recommended in the dCache docs) worked
      - no more hangs due to layout get calls not returning
      - did not fix all issues
      - return of an old issue: nfs kernel threads on clients now hanging waiting for file close calls to return

Nikhef

hanging file close()

- Only seen for file writes
  - kernel logs on client machine fill up with hung process trace backs
  - storage pool logs
    - occasional failed to send rpc error

```
06 Mar 2019 17:05:06 (strijker-03Pool02) [] Failed to send RPC to /2a07:8500:120:e070:0:0:0:79:673 :
Connection reset by peer
```

    - PoolMoverKill and linked java exceptions

```
07 Mar 2019 20:42:47 (strijker-04Pool01) [NFS-hooikuil PoolMoverKill] close called with in-flight read
request
07 Mar 2019 20:42:47 (strijker-04Pool01) [] DSWRITE:
java.nio.channels.ClosedChannelException: null
```

  - nfs door logs

```
07 Mar 2019 12:38:03 (NFS-hooikuil) [] Client reports error NFS4ERR_RETRY_UNCACHED_REP on pool
strijker-04Pool02 for op READ
07 Mar 2019 12:38:03 (NFS-hooikuil) [] Client reports error NFS4ERR_NXIO on pool strijker-04Pool02 for
op READ
07 Mar 2019 12:39:30 (NFS-hooikuil) [] Bad Stateid: op: LAYOUTRETURN : NFS4ERR_BAD_STATEID : State not
known to the client: [5c80f1920000000400001a08, seq: 2]
07 Mar 2019 12:40:11 (NFS-hooikuil) [] Client reports error NFS4ERR_NXIO on pool strijker-04Pool02 for
op WRITE
```

Nikhef

hanging file close()

- nfs clients don't react well to long delays/pauses when transferring files
  - mostly effected writes in stress tests
  - single transfers from multiple clients fine for the 8 nodes used
  - multiple transfers (24 per node) cause problems with 2-3 nodes
  - often a few of the transfers dominate on each node
    - when transferring several large files (1-10 GB) the file close calls can take several hours to return for some files
    - some transfers just fail returning IO errors
      - associated with the PoolMoverKill errors in the pool logs
  - long close delays due to caching by the virtual filesystem
    - with 256 GB per node, just about all writes to dcache can be cache so write() calls return almost immediately
    - nfs mount is done synchronously so for writes return from close() only happens after data is written to disc on the pool node

Nikhef

NFS Client Bottleneck

- single tcp connection per pool to each server from each client
- multiple concurrent transfers managed via a slot table on client
  - default slot table size: centos 6: 16, centos 7: 64
  - each active nfs read/write request assigned a slot
  - not clear (to me) how different processes requests are assigned/compete for slots
  - nfs isn't a block device so the IO schedulers are not available
- nfs module tweaks

```
options nfs max_session_slots=128
options nfs_layout_flexfiles dataserver_retrans=1 dataserver_timeo=150
```

  - increase the slot table size
  - retransmissions and timeouts changes not so important

Nikhef

Fixes/Tweaks

- Health warning
  - fixes/tweaks are a result of
    - googling
    - historic settings on other servers
    - trial and error
    - depressingly small amounts of evidence and genuine understanding
  - this is what worked in our setup
  - not a rigorously methodical investigation: priority to find a working solution
- Server side
  - dcache
    - upgrade to 5.0 (or 5.1 now)
    - use flexfile layouts
    - ensure pool doesn't run out of movers: `mover set max active -queue=regular 10000`
  - IO scheduler
    - tried cfq scheduler as well as default deadline scheduler
    - no change in reliability

Nikhef

Client Settings

- mount options

```
/dcache - fstype=nfs4, intr, minorversion=1, timeo=6000, rsize=32768, wsize=32768
 dcache-door:/dcache
```

  - read/write request sizes important
    - too small < 8k caused problems
    - too large > 128k some problems (but not clear cut)
- previously only tuned network settings on servers, now required on clients
  - fairly standard for high speed nics, contradictory advice for some settings (eg tcp_sack)

```
net.core.netdev_budget: 600
net.core.rmem_default: 524288
net.core.rmem_max: 67108864
net.core.wmem_default: 524288
net.core.wmem_max: 67108864
net.core.optmem_max: 4194304
net.core.somaxconn: 512
net.core.netdev_max_backlog: 250000
net.ipv4.tcp_rmem: "16384 524288 67108864"
net.ipv4.tcp_wmem: "16384 524288 67108864"
net.ipv4.tcp_sack: 1
net.ipv4.tcp_timestamps: 1
```

Nikhef

Client Settings

- filesystem cache tweaks

```
vm.dirty_expire_centisecs: 100
vm.dirty_writeback_centisecs: 50
vm.dirty_background_bytes: 10485760
vm.dirty_bytes: 1073741824
```

  - don’t cache as much in client memory
  - start flushing writes as quickly
  - block new writes until cache empties
    - fakes an IO scheduler (kind of)
    - when filesystem accepts new write after blocking, effectively random which process gets to write data next
    - all processes get some slice of the IO pie

Nikhef

Summary

- Our dcache nfs issues were of our own making
  - didn’t test the new hardware / new OS sufficiently
- Server
  - upgraded to dcache 5.0 (from 3.1)
- Client
  - contention problems when stress testing after server upgrade
  - limited file caching in memory
  - not really happy with solution but it works

Nikhef