NFS Server-Side Copy

Table of Contents

This is a demonstration of how NFS v4 server-side copy ('ssc') (sometimes referred to as copy-offload) works

Server-side copy

Historically, copying a file on an NFS filesystem to another location on an NFS filesystem has involved the client reading the file over NFS from the source filesystem, and then writing those bytes to over NFS to the file on the destination filesystem. Since this requires transferring the bytes twice (once from the source server to the client, and once from the client to the destination server), this is obviously wasteful and slow. This is especially so when the source and destination are on the same NFS server, or even the same filesystem.

To decrease the waste associated with transferring file data twice over the network, and to improve the speed of a file copy operation, 'server-side copy' ('ssc') has been introduced in NFS v4. Server-side copy' refers to two specific modes of operation: 'intra-ssc' and 'inter-ssc'; 'intra-ssc' refers to file copies on the same nfs server, while 'inter-ssc' refers to file copies between different nfs servers. Both modes require client- and server-side support.

Server-side copy is sometimes also referred to as copy-offload.

This demonstration shows 'ssc' as implemented in RHEL 10.2.

Intra-ssc demonstration

With 'intra-ssc', when the client system detects that it is copying a file within an nfs server, rather than reading the file and and writing it again, the client simply opens the source and destination files:

```
 1619   client → server1  NFS 270 V4 Call LOOKUP DH: 0x62d40c52/test_null-intra
 1620  server1 → client   NFS 166 V4 Reply (Call In 1619) LOOKUP Status: NFS4ERR_NOENT
 1621   client → server1  NFS 266 V4 Call LOOKUP DH: 0x62d40c52/test_null
 1622  server1 → client   NFS 402 V4 Reply (Call In 1621) LOOKUP
 1623   client → server1  NFS 326 V4 Call OPEN DH: 0x59258c13/
 1624  server1 → client   NFS 474 V4 Reply (Call In 1623) OPEN StateID: 0x750a
 1625   client → server1  NFS 426 V4 Call OPEN DH: 0x62d40c52/test_null-intra
 1628  server1 → client   NFS 478 V4 Reply (Call In 1625) OPEN StateID: 0x3e90

 1629   client → server1  NFS 298 V4 Call SETATTR FH: 0xab834c72
 1631  server1 → client   NFS 386 V4 Reply (Call In 1629) SETATTR
```

and sends a 'CLONE' operation to the server:

```
 1632   client → server1  NFS 342 V4 Call CLONE Src StateID: 0x2786 Offset: 0 Len: 0
Dst StateID: 0x3b3d Offset: 0
 1633  server1 → client   NFS 254 V4 Reply (Call In 1632) CLONE
```

then closes the source and destination files:

```
 1634   client → server1  NFS 274 V4 Call CLOSE StateID: 0x3e90
 1635  server1 → client   NFS 246 V4 Reply (Call In 1634) CLOSE
 1636   client → server1  NFS 258 V4 Call CLOSE StateID: 0x750a
 1637  server1 → client   NFS 182 V4 Reply (Call In 1636) CLOSE
```

This causes the server to copy the file on the server, without having to transfer any file contents either to or from the client system.

Both server-side and client-side support for 'intra-ssc' was added to the RHEL 8 kernel; client-side support also requires support in 'coreutils' for the 'copy_file_range' system call, also available since RHEL 8.

Inter-ssc demonstration

'Inter-ssc' is more complicated than 'intra-ssc', and server-side support is only available beginning with RHEL 10.2. With 'inter-ssc', the client opens the source and destination files:

```
    1      client → server1      NFS 418 V4 Call (Reply In 3) OPEN DH:
0x62d40c52/testfile
    3     server1 → client       NFS 538 V4 Reply (Call In 1) OPEN StateID: 0xafa9

  186      client → server2      NFS 270 V4 Call (Reply In 187) LOOKUP DH:
0x62d40c52/testfile-copy
  187     server2 → client       NFS 166 V4 Reply (Call In 186) LOOKUP Status:
NFS4ERR_NOENT
  192      client → server2      NFS 426 V4 Call (Reply In 194) OPEN DH:
```

```
0x62d40c52/testfile-copy
  194     server2 → client       NFS 538 V4 Reply (Call In 192) OPEN StateID: 0xafa9

  195      client → server2      NFS 298 V4 Call (Reply In 197) SETATTR FH: 0xb6cce673
  197     server2 → client       NFS 346 V4 Reply (Call In 195) SETATTR
```

next, the client informs the source server it will be making a 'COPY' call to another system, and to expect a connection:

```
  198      client → server1      NFS 290 V4 Call (Reply In 199) COPY_NOTIFY StateID:
0x3fff
        Opcode: COPY_NOTIFY (61)
            StateID
                [StateID Hash: 0x3fff]
                StateID seqid: 0
                StateID Other: 91894069328b834104000000
                [StateID Other hash: 0xd1555989]
            netloc type: NL4_NETADDR (3)
            netaddr
                r_netid: tcp
                    length: 3
                    contents: tcp
                    fill bytes: opaque data
                r_addr: 192.168.122.74.8.1
                    length: 18
                    contents: 192.168.122.74.8.1
                    fill bytes: opaque data
                [IPv4 address 192.168.122.74, protocol=tcp, port=2049]
```

the source server replies with connection information which the destination server should use for connecting in:

```
  199     server1 → client       NFS 234 V4 Reply (Call In 198) COPY_NOTIFY
        Opcode: COPY_NOTIFY (61)
            Status: NFS4_OK (0)
            seconds: 90
            nseconds: 0
            StateID
                [StateID Hash: 0x5767]
                StateID seqid: 1
                StateID Other: 918940698f8a834101000000
                [StateID Other hash: 0x0d0f6179]
            Source Server count: 1
                Source Server: 0
                    netloc type: NL4_NETADDR (3)
                    netaddr
                        r_netid: tcp
                            length: 3
```

```
                            contents: tcp
                            fill bytes: opaque data
                        r_addr: 192.168.122.75.8.1
                            length: 18
                            contents: 192.168.122.75.8.1
                            fill bytes: opaque data
                        [IPv4 address 192.168.122.75, protocol=tcp, port=2049]
```

the client then sends a 'COPY' instruction to the destination server; the 'COPY' opcode contains information necessary to tell the destination server how to connect to the source server for the information:

```
  201      client → server2      NFS 374 V4 Call (Reply In 496) COPY Src StateID:
0x5767 Offset: 0 Len: 5242880 Dst StateID: 0xfb34 Offset: 0
        Opcode: COPY (60)
            StateID
                [StateID Hash: 0x5767]
                StateID seqid: 1
                StateID Other: 918940698f8a834101000000
                [StateID Other hash: 0x0d0f6179]
            StateID
                [StateID Hash: 0xfb34]
                StateID seqid: 0
                StateID Other: 8e894069ee7032e904000000
                [StateID Other hash: 0x09d332f3]
            offset: 0
            length: 5242880
            copy consecutively?: Yes
            copy synchronous?: No
            Source Server count: 1
                Source Server: 0
                    netloc type: NL4_NETADDR (3)
                    netaddr
                        r_netid: tcp
                            length: 3
                            contents: tcp
                            fill bytes: opaque data
                        r_addr: 192.168.122.75.8.1
                            length: 18
                            contents: 192.168.122.75.8.1
                            fill bytes: opaque data
                        [IPv4 address 192.168.122.75, protocol=tcp, port=2049]
```

At this point, the destination server establishes a network connection and NFS session with the source server (essentially mounting the source server):

```
  203     server2 → server1      TCP 74 681 → 2049 [SYN] Seq=0 Win=64240 Len=0
MSS=1460 SACK_PERM TSval=3420159675 TSecr=0 WS=128
```

```
  204     server1 → server2      TCP 74 2049 → 681 [SYN, ACK] Seq=0 Ack=1 Win=65160
Len=0 MSS=1460 SACK_PERM TSval=1466131418 TSecr=3420159675 WS=128
  205     server2 → server1      TCP 66 681 → 2049 [ACK] Seq=1 Ack=1 Win=64256 Len=0
TSval=3420159676 TSecr=1466131418

  210     server2 → server1      NFS 358 V4 Call (Reply In 211) EXCHANGE_ID
  211     server1 → server2      NFS 326 V4 Reply (Call In 210) EXCHANGE_ID
  212     server2 → server1      NFS 358 V4 Call (Reply In 213) EXCHANGE_ID
  213     server1 → server2      NFS 326 V4 Reply (Call In 212) EXCHANGE_ID
  214     server2 → server1      NFS 298 V4 Call (Reply In 215) CREATE_SESSION
  215     server1 → server2      NFS 194 V4 Reply (Call In 214) CREATE_SESSION
  216     server2 → server1      NFS 210 V4 Call (Reply In 218) RECLAIM_COMPLETE
  217     server1 → server2      TCP 66 2049 → 681 [ACK] Seq=677 Ack=1005 Win=64384
Len=0 TSval=1466131469 TSecr=3420159685
  218     server1 → server2      NFS 158 V4 Reply (Call In 216) RECLAIM_COMPLETE
  219     server2 → server1      NFS 226 V4 Call (Reply In 221) PUTROOTFH | GETATTR
  220     server1 → server2      TCP 66 2049 → 681 [ACK] Seq=769 Ack=1165 Win=64256
Len=0 TSval=1466131508 TSecr=3420159765
  221     server1 → server2      NFS 234 V4 Reply (Call In 219) PUTROOTFH | GETATTR
...
```

And the destination server begins making READ requests to the source server:

```
  238     server2 → server1      NFS 262 V4 Call (Reply In 253) READ StateID: 0x52ca
Offset: 0 Len: 131072
  253     server1 → server2      NFS 29886 V4 Reply (Call In 238) READ
  255     server2 → server1      NFS 262 V4 Call (Reply In 260) READ StateID: 0x52ca
Offset: 131072 Len: 131072
  256     server2 → server1      NFS 262 V4 Call (Reply In 264) READ StateID: 0x52ca
Offset: 262144 Len: 131072
...
```

After the destination server has completed making 'READ' calls to the source server, it responds to the client's 'COPY' operation:

```
  496     server2 → client       NFS 230 V4 Reply (Call In 201) COPY
```

There are also several callbacks from the nfs servers to the client; the source server asks the client the file size:

```
  497     server1 → client       NFS CB 246 V1 CB_COMPOUND Call (Reply In 499) <EMPTY>
CB_SEQUENCE;CB_GETATTR
    Operations (count: 2)
        Opcode: CB_SEQUENCE (11)
        Opcode: CB_GETATTR (3)
        reqd_attr: Change (3)
        reqd_attr: Size (4)
```

```
  499      client → server1      NFS CB 190 V1 CB_COMPOUND Reply (Call In 497) <EMPTY>
CB_SEQUENCE;CB_GETATTR
    Operations (count: 2)
        Opcode: CB_SEQUENCE (11)
        Opcode: CB_GETATTR (3)
                    changeid: 7584498875976546347
                    size: 5242880
```

And the destination server informs the client that the copy offload is complete:

```
  502     server2 → client       NFS CB 310 V1 CB_COMPOUND Call (Reply In 504) <EMPTY>
CB_SEQUENCE;CB_OFFLOAD
        Opcode: CB_SEQUENCE (11)
        Opcode: CB_OFFLOAD (15)
            FileHandle
            StateID
            Status: NFS4_OK (0)
            Callback StateIds: 0
            length: 5242880
            committed: FILE_SYNC4 (2)
  504      client → server2      NFS CB 154 V1 CB_COMPOUND Reply (Call In 502) <EMPTY>
CB_SEQUENCE;CB_OFFLOAD
```

Getting started

On the server, 'intra-ssc' is enabled automatically, however 'intra-ssc' must be specifically enabled through a kernel module parameter for the 'nfsd' module:

```
# modinfo nfsd
parm:           inter_copy_offload_enable:Enable inter server to server copy offload.
Default: false (inter_copy_offload)
```

'inter-ssc' also has a second module parameter:

```
parm:           nfsd4_ssc_umount_timeout:idle msecs before unmount export from source
server (int)
```

Because 'intra-ssc' results in the destination server mounting the filesystem from the source server, the filesystem will remain mounted on the destination server for a time after the copy completes. Every 'nfsd4_ssc_umount_timeout' msec, the destination server will check how long the mount has been idle, and if it has been idle for longer than 'nfsd4_ssc_umount_timeout', 'nfsd' will unmount the filesystem. This means that the filesystem may remain mounted for nearly 2x the mount timeout parameter before unmount occurs. The default value is 900000, which is 15 minutes.

Note that although the filesystem will be mounted, and a TCP connection will remain ESTABLISHED

between the source and destination nfs servers, the server-to-server mount will not appear in the mount tables in /proc for any process. The filesystem will appear in the '/proc/fs/nfsfs/servers' and '/proc/fs/nfsfs/volumes' on the destination server, and 'rpc_xprt' and 'rpc_clnt' directories will be present in '/sys/kernel/debug/sunrpc'.

To enable 'inter-ssc' and set the unmount timeout parameter, edit the '/etc/modprobe.d/nfsd.conf' file, and set the parameters:

```
options nfsd inter_copy_offload_enable=Y nfsd4_ssc_umount_timeout=30000
```

(the above will enable 'inter-ssc' and set the unmount timeout to 30 seconds)

When it's not working

intra-ssc is not supported by the nfs server

For 'intra-ssc', if the nfs server does not support the feature, the 'CLONE' attempt will return 'NFS4ERR_NOTSUPP', the client will mark the server as not supporting 'ssc', and the client will fall back to 'READ' + 'WRITE' operations:

```
    4          client → server-rhel7  NFS 266 V4 Call (Reply In 5) LOOKUP DH:
0x62d40c52/testfile-copy
    5    server-rhel7 → client        NFS 166 V4 Reply (Call In 4) LOOKUP Status:
NFS4ERR_NOENT
    6          client → server-rhel7  NFS 258 V4 Call (Reply In 7) LOOKUP DH:
0x62d40c52/testfile
    7    server-rhel7 → client        NFS 350 V4 Reply (Call In 6) LOOKUP
    8          client → server-rhel7  NFS 322 V4 Call (Reply In 9) OPEN DH:
0xe4e4d2b2/
    9    server-rhel7 → client        NFS 422 V4 Reply (Call In 8) OPEN StateID:
0x1e1b
   10          client → server-rhel7  NFS 374 V4 Call (Reply In 11) OPEN DH:
0x62d40c52/testfile-copy
   11    server-rhel7 → client        NFS 426 V4 Reply (Call In 10) OPEN StateID:
0x276d
   12          client → server-rhel7  NFS 294 V4 Call (Reply In 13) SETATTR FH:
0x7b425c64
   13    server-rhel7 → client        NFS 334 V4 Reply (Call In 12) SETATTR

   14          client → server-rhel7  NFS 342 V4 Call (Reply In 15) CLONE Src StateID:
0x070d Offset: 0 Len: 0 Dst StateID: 0x22c0 Offset: 0
   15    server-rhel7 → client        NFS 182 V4 Reply (Call In 14) CLONE Status:
NFS4ERR_NOTSUPP

   16          client → server-rhel7  NFS 338 V4 Call (Reply In 17) COPY Src StateID:
0x070d Offset: 0 Len: 5242880 Dst StateID: 0x22c0 Offset: 0
   17    server-rhel7 → client        NFS 182 V4 Reply (Call In 16) COPY Status:
```

```
NFS4ERR_NOTSUPP

   18          client → server-rhel7  NFS 266 V4 Call (Reply In 37) READ StateID:
0x070d Offset: 0 Len: 131072
```

The client won’t retry the 'CLONE' again, since it marked the feature unavailable during the first failure.

inter-ssc where the source server does not support the feature

For 'inter-ssc', if the source server does not support the feature:

```
    4          client → server-rhel7  NFS 258 V4 Call (Reply In 5) LOOKUP DH:
0x62d40c52/testfile
    5    server-rhel7 → client        NFS 350 V4 Reply (Call In 4) LOOKUP
    6          client → server-rhel7  NFS 322 V4 Call (Reply In 7) OPEN DH:
0xe4e4d2b2/
    7    server-rhel7 → client        NFS 422 V4 Reply (Call In 6) OPEN StateID:
0x03aa
    8          client → server1       NFS 426 V4 Call (Reply In 11) OPEN DH:
0x62d40c52/testfile-copy
   11         server1 → client        NFS 538 V4 Reply (Call In 8) OPEN StateID:
0xafa9
   13          client → server1       NFS 298 V4 Call (Reply In 15) SETATTR FH:
0x466e17e3
   15         server1 → client        NFS 346 V4 Reply (Call In 13) SETATTR
   16          client → server-rhel7  NFS 290 V4 Call (Reply In 17) COPY_NOTIFY
StateID: 0x1abc
   17    server-rhel7 → client        NFS 166 V4 Reply (Call In 16) COPY_NOTIFY
Status: NFS4ERR_NOTSUPP

   19          client → server-rhel7  NFS 266 V4 Call (Reply In 41) READ StateID:
0x1abc Offset: 0 Len: 131072
```

Again, the client won’t retry the 'COPY_NOTIFY', since it marked the feature unavailable during the first failure.

inter-ssc where the destination server does not support the feature

For 'inter-ssc' where the destination server does not support the feature:

```
    7        client → server-rhel7 NFS 266 V4 Call (Reply In 8) LOOKUP DH:
0x62d40c52/testfile-copy
    8  server-rhel7 → client       NFS 166 V4 Reply (Call In 7) LOOKUP Status:
```

```
NFS4ERR_NOENT
    9        client → server1      NFS 262 V4 Call (Reply In 10) LOOKUP DH:
0x62d40c52/testfile
   10       server1 → client       NFS 414 V4 Reply (Call In 9) LOOKUP
   12        client → server1      NFS 326 V4 Call (Reply In 13) OPEN DH: 0xa7b25520/
   13       server1 → client       NFS 486 V4 Reply (Call In 12) OPEN StateID: 0xafa9
   14        client → server-rhel7 NFS 374 V4 Call (Reply In 15) OPEN DH:
0x62d40c52/testfile-copy
   15  server-rhel7 → client       NFS 426 V4 Reply (Call In 14) OPEN StateID: 0x703c
   16        client → server-rhel7 NFS 294 V4 Call (Reply In 17) SETATTR FH:
0xa31bca89
   17  server-rhel7 → client       NFS 334 V4 Reply (Call In 16) SETATTR

   18        client → server1      NFS 290 V4 Call (Reply In 19) COPY_NOTIFY StateID:
0xc528
   19       server1 → client       NFS 234 V4 Reply (Call In 18) COPY_NOTIFY

   20        client → server-rhel7 NFS 374 V4 Call (Reply In 21) COPY Src StateID:
0xae04 Offset: 0 Len: 5242880 Dst StateID: 0x7591 Offset: 0
   21  server-rhel7 → client       NFS 158 V4 Reply (Call In 20) SEQUENCE | PUTFH
Status: NFS4ERR_STALE

   22        client → server1      NFS 254 V4 Call (Reply In 23) OFFLOAD_CANCEL
StateID: 0xae04
   23       server1 → client       NFS 166 V4 Reply (Call In 22) OFFLOAD_CANCEL
   24        client → server1      NFS 266 V4 Call (Reply In 41) READ StateID: 0xc528
Offset: 0 Len: 131072
```

Here, the client makes the 'COPY_NOTIFY' call to the source server, then attempts the 'COPY' operation with the destination server. In this case, the destination server does not recognize the filehandle, since the filehandle is not actually on the destination; it's on the source. So, the destination server replies with 'NFS4ERR_STALE'. The client then cancels the copy offload began with the 'COPY_NOTIFY', reads the file from the source, and writes it to the destination.

In this case, the client will likely retry the copy offload, since 'NFS4ERR_STALE' did not indicate a lack of support for the feature. But the 'COPY' will continue to fail, and the client will fall back to the 'READ' + 'WRITE' behavior.

inter-ssc without the source filesystem being exported to the destination server

For 'inter-ssc', the destination server **must** be able to mount the source server, otherwise the copy offload will not be permitted. In other words, the filesystem to mount must be exported to the destination server.

Here is what this failure looks like:

```
   71    client → server2   NFS 270 V4 Call (Reply In 72) LOOKUP DH:
```

```
0x62d40c52/testfile-copy
   72   server2 → client    NFS 166 V4 Reply (Call In 71) LOOKUP Status: NFS4ERR_NOENT
   73    client → server1   NFS 246 V4 Call (Reply In 74) ACCESS FH: 0x62d40c52,
[Check: RD LU MD XT DL XAR XAW XAL]
   74   server1 → client    NFS 238 V4 Reply (Call In 73) ACCESS, [Allowed: RD LU MD
XT DL XAR XAW XAL]

   76    client → server1   NFS 262 V4 Call (Reply In 77) LOOKUP DH:
0x62d40c52/testfile
   77   server1 → client    NFS 414 V4 Reply (Call In 76) LOOKUP
   78    client → server1   NFS 326 V4 Call (Reply In 79) OPEN DH: 0xa7b25520/
   79   server1 → client    NFS 486 V4 Reply (Call In 78) OPEN StateID: 0xafa9
   80    client → server2   NFS 426 V4 Call (Reply In 81) OPEN DH:
0x62d40c52/testfile-copy
   81   server2 → client    NFS 538 V4 Reply (Call In 80) OPEN StateID: 0xafa9
   82    client → server2   NFS 298 V4 Call (Reply In 83) SETATTR FH: 0x8fc86f57
   83   server2 → client    NFS 346 V4 Reply (Call In 82) SETATTR

   84    client → server1   NFS 290 V4 Call (Reply In 85) COPY_NOTIFY StateID: 0xeac2
   85   server1 → client    NFS 234 V4 Reply (Call In 84) COPY_NOTIFY

   86    client → server2   NFS 374 V4 Call (Reply In 87) COPY Src StateID: 0xdce8
Offset: 0 Len: 5242880 Dst StateID: 0xd0b4 Offset: 0
```

here, the destination server will attempt the mount, be denied access, and return an error:

```
   87   server2 → client    NFS 182 V4 Reply (Call In 86) COPY Status:
NFS4ERR_OFFLOAD_DENIED

   88    client → server1   NFS 254 V4 Call (Reply In 89) OFFLOAD_CANCEL StateID:
0xdce8
   89   server1 → client    NFS 166 V4 Reply (Call In 88) OFFLOAD_CANCEL

   90    client → server1   NFS 266 V4 Call (Reply In 106) READ StateID: 0xeac2
Offset: 0 Len: 131072
   ...
```

Because permission was denied, the client again falls back to 'READ' + 'WRITE'. Here, the destination server is not marked as not supporting the feature, since it was a permission problem, not a lack of support for the feature.

Conclusions from the demonstration

'server-side copy' can reduce data transfer and improve file copy performance, but there must be client-side and server-side support, and the client's 'cp' must also support the 'copy_file_range' system call. 'Intra-ssc' is available in the latest RHEL 8, and 'inter-ssc' is available on nfs servers beginning with RHEL 10.2.