Project Status

NFSv4 Extensions for Performance and Interoperability

Center for Information Technology Integration

This is a report on the status of CITI's EMC-funded pNFS development project as of September 4, 2008. Items marked in blue reflect change from the June 19, 2008 report.

Sessions in the generic Linux pNFS client

Task	Description	Status
S1	Session recovery.	This task is complete.
S2	Callback channel.	This task is complete.
S3	NFSv4.1 back channel security using machine credentials.	To provide for back channel security, we added support for machine credentials in the SETCLIENTID call. This makes it possible for the callback client to establish a secure channel to the corresponding principal on the callback server. Patches were committed to Linux 2.6.26-rc1. Now we are working on extending the RPC upcall mechanism so that the callback client can acquire appropriate credentials from gssd. Patches were posted to the linux-nfs mailing list and are under discussion.
S4	NFSv4.1 back channel security using secret state verifiers.	No progress to report

Other generic pNFS client issues

Task	Description	Status
C1	LAYOUTGET, LAYOUTRETURN, and CB_LAYOUTRECALL.	LAYOUTGET and LAYOUTRETURN are complete. We have a general framework and an untested draft implementation of CB_LAYOUTRECALL, with testing still to come.
C2	CB_RECALL_ANY, RECLAIM_COMPLETE, and CB_RECALLABLE_OBJ_AVAIL.	No progress to report. (So far, the NFSv4.1 development community is deferring work on these non-critical elements.)
C3	Integration of block layout requirements into generic client.	This task is under way and ongoing. The main pNFS branch now includes appropriate hooks for the block driver in the write path.
C4	Implement new NFSv4.1 draft 19–21 pNFS features and behavior.	Layout stateid is under active development in the NFSv4.1 development community, with Andy Adamson (NetApp) leading the way. See Appendix for status. Device notification is under active development in the pNFS development community, with Marc Eshel (IBM) leading the development activity. Draft rewrites have simplified this task considerably by eliminating the ADD operation. XDR formats have been worked out and we have an initial implementation of the generic client and server processing code.
C5	Reboot recovery.	This task is nearly complete.

Block layout module

Task	Description	Status
B1	Rebase the implementation from block draft 3 to block draft 6.	We have rebased to draft 9.
B2	Extend the block layout implementation to support large server block sizes.	This task is complete.
B3	Block layout client implementation based on architectural review.	Based on the architectural review, we need to move disk scanning out of the kernel and into user space. Tang Haiying is working on this task.
B4	Support for complex volume topologies using the Linux device mapper (dm) needs to be reviewed to meet performance and quality requirements.	We have a working implementation that needs further testing. When we return to task B3, we will revisit this implementation.
B5	Extend the layout cache implementation to support at least two devices.	We have a working implementation that needs further testing. When we return to task B3, we will update this implementation.
B6	Extend the device mapper to support the asynchronous CB_NOTIFY_DEVICEID callback operation.	No progress to report. Block-specific device notification depends on generic device notification (Task C4). We will begin work on this task soon.
B7	The block layout client must implement a timed lease I/O fencing mechanism to insulate against network partition.	No progress to report

PyNFS

Task	Description	Status
P1	Update PyNFS client and server to support new protocol features in the latest drafts.	The PyNFS client and server now support the latest drafts (minorversion1 draft 23 and pnfs-block draft 9).
P2	Enhance the block server implementation to pass full Connectathon tests.	The PyNFS server passes all Connectathon NFSv4 and non-pNFS NFSv4.1 tests except for the large file test. We now have a prototype implementation of a "real" file system that supports read, write, and file creation.

Milestone summary

The following tasks were projected to be complete by the May 2008 Connectathon.

Task	Description	Status
S1	Session recovery	**Complete**
S2	Callback channel implementation	**Complete**
B1	Block layout draft 6	**Complete**
B2	Server block sizes greater then 4 KB	**Complete**
B3	Revisit block layout client implementation based on architectural review	**Under way**

The following tasks are projected to be complete by the Fall 2008 Bakeathon.

Task	Description	Status
S3	Back channel security using machine credentials	**No progress**
C1	LAYOUTGET, LAYOUTRETURN, and CB_LAYOUTRECALL	**Nearly complete**
C2	CB_RECALL_ANY, RECLAIM_COMPLETE, CB_RECALLABLE_OBJ_AVAIL	**No progress**
P1	PyNFS block client and server support latest drafts	**Complete**
P2	PyNFS block server passes full Connectathon tests, prototype file system.	**Nearly complete**

The following tasks are projected to be under way by the Fall 2008 Bakeathon.

Task	Description	Status
C3	Integration of block layout requirements into the generic client	**Under way**
C4	Draft 19–21 pNFS features and behavior. See Appendix for status.	**Under way**
B4	Complex volume topologies	**Testing**
B5	Copy-on-write	**Testing**

The remaining tasks are projected to be complete by the end of the project.

Task	Description	Status
S4	NFSv4.1 back channel security using secret state verifiers	**No progress**
C5	Reboot recovery	**Nearly complete**
B6	CB_NOTIFY_DEVICEID	**No progress**
B7	Timed lease I/O fencing mechanism	**No progress**

Appendix: Notes on reboot recovery and layout stateid

Andy Adamson (NetApp), who leads the generic pNFS development, provides the following status information.

- Session recovery, clientid recovery, and session sync / async error recovery, and network partition recovery are functional in the Linux generic NFSv4.1 implementation.
- SEQUENCE operation status reply, the TEST_STATEID operation, the FREE_STATEID operation, and the RECLAIM_COMPLETE operations are not yet supported.
- The pNFS client responds to all SEQUENCE operation errors by destroying the session and creating a new one.
- Metadata server / data server reboot recovery is being tested.
- NFSv4.1 and pNFS reboot recovery are expected to be functioning in time for interoperability testing at the Austin Bake-a-thon.
- The following tables indicate the status of layout stateid in the Linux generic pNFS implementation. Testing indicates compatibility with the Solaris implementation.

Server issues

Reference	Narrative	Status
§12.5.2 ¶2	When a client has no layout on a file, it MUST present a stateid as returned by OPEN, a delegation stateid, or a byte-range lock stateid in the loga_stateid argument. The first successful LAYOUTGET processed by the server using a non-layout stateid as an argument MUST have the "seqid" field of the layout stateid in the response set to one. Thereafter … the "seqid" MUST NOT be set to zero.	Complete
§12.5.3 ¶1	Once a layout stateid is changed, the "other" field will stay constant unless the stateid is revoked, or the client returns all layouts on the file and the server disposes of the stateid. After the layout stateid is established, the server increments by one the value of the "seqid" in each subsequent LAYOUTGET and LAYOUTRETURN response, and in each CB_LAYOUTRECALL request.	Complete
§12.5.3 ¶5	Once a client has no more layouts on a file, the layout stateid is no longer valid, and MUST NOT be used. Any attempt to use such a layout stateid will result in NFS4ERR_BAD_STATEID.	Complete
§12.5.5.2	It is the server's responsibility to avoid inconsistencies regarding the layouts provided.	In test
§12.5.5.2.1.1 ¶3	It is permissible for the client to use the current stateid for LAYOUTGET operations for example when compounding LAYOUTGETs or compounding OPEN and LAYOUTGETs. It is also permissible to use the current stateid when compounding LAYOUTRETURNs.	Incomplete
§12.5.5.2.1.1 ¶4	It is permissible for the client to use the current stateid when combining LAYOUTRETURN and LAYOUTGET operations for the same file in the same COMPOUND request since the server MUST process these in order.	Incomplete

Reference	Narrative	Status
§12.5.5.2.1.3	1. The client sent the LAYOUTGET before processing the CB_LAYOUTRECALL. The "seqid" in the layout stateid of LAYOUTGET is two less than the "seqid" in CB_LAYOUTRECALL. The server returns NFS4ERR_RECALLCONFLICT to the client, which indicates to the client that there is a pending recall. 2. The client sent the LAYOUTGET after processing the CB_LAYOUTRECALL, but the LAYOUTGET arrived before the LAYOUTRETURN and the response to CB_LAYOUTRECALL that completed that processing. The "seqid" in the layout stateid of LAYOUTGET is equal to or greater than that of the "seqid" in CB_LAYOUTRECALL. The server has not received a response to the CB_LAYOUTRECALL, so it returns NFS4ERR_RECALLCONFLICT. 3. The client sent the LAYOUTGET after processing the CB_LAYOUTRECALL, the server received the CB_LAYOUTRECALL response, but the LAYOUTGET arrived before the LAYOUTRETURN that completed that processing. The "seqid" in the layout stateid of LAYOUTGET is equal to that of the "seqid" in CB_LAYOUTRECALL. The server has received a response to the CB_LAYOUTRECALL, so it returns NFS4ERR_RETURNCONFLICT.	Incomplete

Client issues

Reference	Narrative	Status
§12.5.2 ¶2	When a client has no layout on a file, it MUST present a stateid as returned by OPEN, a delegation stateid, or a byte-range lock stateid in the loga_stateid argument.	Complete
§12.5.3 ¶2	The "seqid" value is used by the client to properly sort responses to LAYOUTGET and LAYOUTRETURN. The "seqid" is also used to prevent race conditions between LAYOUTGET and CB_LAYOUTRECALL.	Incomplete
§12.5.3 ¶3	Once the client receives a layout stateid, it MUST use the correct "seqid" for subsequent LAYOUTGET or LAYOUTRETURN operations. The correct "seqid" is defined as the highest "seqid" value from responses of fully processed LAYOUTGET or LAYOUTRETURN operations or arguments of a fully processed CB_LAYOUTRECALL operation.	In progress
§12.5.3 ¶3	In the case of overlapping layout ranges, the ordering information will provide the client the knowledge of which layout ranges are held.	Incomplete
§12.5.3 ¶4	LAYOUTGET results may be processed in parallel. LAYOUTRETURN results may be processed in parallel. LAYOUTGET and LAYOUTRETURN responses may be processed in parallel as long as the ranges do not overlap. CB_LAYOUTRECALL request processing MUST be processed in "seqid" order at all times.	Incomplete
§12.5.3 ¶4	For LAYOUTGET results, if the client is not using the forgetful model, it MUST first update its record of what ranges of the file's layout it has before using the seqid. For LAYOUTRETURN results, the client MUST delete the range from its record of what ranges of the file's layout it had before using the seqid.	Complete
§12.5.3 ¶4	For CB_LAYOUTRECALL arguments, the client MUST send a response to the recall before using the seqid.	Depends on §12.5.5.2.1.1

Reference	Narrative	Status
§12.5.5.2.1 ¶1	A client MUST NOT process a CB_LAYOUTRECALL that implies one or more outstanding LAYOUTGET or LAYOUTRETURN operations to which the client has not yet received a reply. The client MUST wait before processing such a CB_LAYOUTRECALL until it processes all replies for outstanding LAYOUTGET and LAYOUTRETURN operations for the corresponding file with seqid less than the seqid given by CB_LAYOUTRECALL.	Incomplete
§12.5.5.2.1.1 ¶2	It is permissible for the client to send in parallel multiple LAYOUTGET operations for the same file or multiple LAYOUTRETURN operations for the same file, and a mix of both.	Complete
§12.5.5.2.1.1 ¶4	It is permissible for the client to use the current stateid when combining LAYOUTRETURN and LAYOUTGET operations for the same file in the same COMPOUND request since the server MUST process these in order. However, if a client does send such COMPOUND requests, it MUST NOT have more than one outstanding for the same file at the same time and MUST NOT have other LAYOUTGET or LAYOUTRETURN operations outstanding at the same time for that same file.	Incomplete
§12.5.5.2.1.2	1. The server processed the LAYOUTGET before issuing the recall, so the LAYOUTGET must be waited for because it may be carrying layout information that will need to be returned to deal with the CB_LAYOUTRECALL. The client knows it needs to wait for the LAYOUTGET response before processing the recall (or the client can return NFS4ERR_DELAY). 2. The server sent the callback before receiving the LAYOUTGET. The server will not respond to the LAYOUTGET until the CB_LAYOUTRECALL is processed.	Incomplete