10/30/90

X3T9.2/90-173

FROM: Mike Peper
Hewlett-Packard

To: X3T9.2 Members

Re: Multiple I/O Processes in SCSI

The following is a note from a colleague expressing his view that the SCSI-2 spec means to disallow multiple active I/O Processes. I do not necessarily share all of his views, but I do think he is quite correct in stating that the spec is very vague and somewhat contradictory on the subject.

Rather than go the full RFI route, I would appreciate it if you could read this over, form an opinion (something I think we all are quite good at!), and let me know how you feel when my agenda item comes up. I would hope that eventually we could clean up this wording at some point to clarify exactly what the committee intends to support.

Thanks!

I believe the proposal to gain performance through multiple active IOPs (on a LUN) has a fundamental flaw in that it violates the SCSI-2 spec. It is my interpetation that multiple active IOPs are not legal. I believe the text of the SCSI-2 spec. supports this view.

In the following I will attempt to justify my belief:

First some definitions from the SCSI-2 glossary

active I/O process. An I/O process that is presently in execution (not queued).

current I/O process. The I/O process that is presently connected on the SCSI bus.

I/O process. An I/O process consists of one initial connection and zero or more reconnections, all pertaining to a single command or a group of linked commands. More specifically, the connection(s) pertain to a nexus as defined below in which zero or more command descriptor blocks are transferred. An I/O process begins with the establishment of a nexus. An I/O process normally ends with the BUS FREE phase following sucessful transfer of a COMMAND COMPLETE or a RELEASE RECOVERY message. An I/O process also ends with the BUS FREE phase following an ABORT, ABORT TAG, BUS DEVICE RESET, or CLEAR QUEUE message or when a hard RESET condition or an unexpected disconnect occurs.

queued I/O process. An I/O process that is in the command queue and has not begun execution.

In 5.6.1 on the ABORT message

"The abort message is send from the initiator to the target to clear the active I/O process plus any queued I/O process for the I_T_x nexus. ... The pending data, status, and queued I/O processes for any other I_T_x nexus shall not be cleared."

Note the use of the definite article "the" in describing the(an) active I/O process. Also note the explicit description to leave things alone for other I_T_x's. It mentions queued I/O processes but makes no mention of what to do with another active IOP for this I_T_x.

In 5.6.2 on ABORT TAG

"Pending status, data, and commands for other active or queued I/O processes shall not be affected. Execution of other I/O processes queued for the I_T_x nexus shall not be aborted."

See 2nd half of comment above.

In 5.6.4 on CLEAR QUEUE

Here I find the 1st of 2 supporting references of multiple active I/O processes.

"All active I/O processes shall be terminated. The medium may have been altered by partially executed commands."

Plural "processes" and "commands" surely suggested multiple active I/O processes.

In 5.6.22 on TERMINATE I/O PROCESS

The 2nd reference supporting multiple active IOP's.

"The TERMINATE I/O PROCESS message shall not affect pending status, data, and commands for other queued or executing I/O processes.

Another plural "processes". The reference to executing (which is only indirectly defined) vs. active seems inappropriate.

IF the 2 preceding reference were the only references supporting multiple active IOPs then I wonder if it is an English oversight?

In 6.6 on Contingent Allegiance Condition

"Execution of queued commands for the logical unit or target routine for which the contingent allegiance condition exists shall be suspended until the contingent allegiance condition is cleared."

Note there is no discussion of suspending other active IOP's. I believe an initiator must know that any outstanding IOPs are queued and not active to be able to take corrective action.

In 6.7 on Extended Contingent Allegiance Condition

"Execution of queued commands for the logical unit for which the extended contingent allegiance condition exists shall be suspended until the contingent allegiance condition is cleared."

Same comment as on 6.6

In 6.8 on Queued I/O Processes

"Queuing of I/O proccesses allows a target to accept multiple I/O processes for execution at a later time."

This sentence might have said "immediate or deferred execution".

In 6.8.1 on Untagged Queuing

"Untagged queuing allows a target to accept a command from an initiator for a logical unit or a target routine while a command from another initiator is being executed."

"while a command" not "while commands"

"If the disconnect privilege is not granted in the IDENTIFY message of the new I/O process, the target may either suspend the active I/O process or it may return BUSY status to the new I/O process."

Another definite article "the active I/O process"

In 6.8.2 on Tagged Queuing

"When adding an I/O process, the initiator may specify fixed order of execution, allow the target to define the order of exectuion, or specify that the I/O process is the be executed next."

"If only SIMPLE QUEUE TAG messages are used, the target may execute the command in any order that is deemed desirable within the constraints of the queue management algorithm specified in the control mode page (see 7.3.3.1)"

"A command received with a HEAD OF QUEUE TAG message is placed first in the queue, to be executed next."

These are very strong statements against multiple active IOP's. It is at the very heart of the definition of tagged queuing. It defines the degrees of freedoms the target has respect to execution order. Notice the phrases "executed next", "order of execution" and "in any order".

Some more from the section:

"The first post_recovery option is to continue execution of commands in the queue after the contingent allegiance or extended allegiance condition has cleared. ... During this time all commands in the queue are suspended."

"The second recovery option clears the queue after the contingent allegiance or extended contingent allegiance condition has been cleared."

What happens to any other active IOP's?

Now in this section is the following interesting paragraph:

"If commands are combined by the queuing algorithm such that the error affects more than more command, then a contingent allegiance condition shall be generated for all affected commands."

The argument might be made that this statement supports multiple active IOP's and I considered this thought. But after looking at this statement a while, I came to the conclusion that in fact it supports a single active IOP (ordered execution) model. It in effect states that you need to produce n sequential contingent allegiance conditions, thus hiding any optimization the target may have done.

Let me make a distinction here between multiple active IOP's and target optimization. I am not suggesting that a target can not do some parallel processing on IOP's. Just that any optimization it chooses to do must not be visible to the host functionally. Unfortunately multiple active IOP's are visible.

In 7.3.1.1 on Control Mode Page

"A queue error management (QErr) bit of zero specifies that those commands still queued after the target has entered the contingent allegiance or extended contingent allegiance conditions shall continue execution in a normal manner when that condition has terminated (see 6.8) A QErr bit of one specifies that those commands still queued after the target has entered the contingent allegiance or extended contingent allegiance conditions shall be aborted when that condition has terminated."

If multiple active IOP are possible then these paragraphs leave alot to be said about error recovery!

Appendix E. Data Integrity and I/O Process Queuing

This appendix illustrates one possible implementation of command queuing. It discusses legal SCSI-2 command ordering and re-ordering algorithms. The concept of order is predominant throughout the appendix.

These are the major areas of the spec that do not make sense with the concept of multiple I/O Processes being active. I am very concerned about initiators being capable of handling a target which might use this method.