Exception Recovery for Non-Interlocked Interconnects

Charles Monia
Digital Equipment Corporation
November 5, 1996
X3T10/95-352R3
In a non-interlocked bus, the potential for out-of-order command execution exists whenever:

- More than one command may be in transit at any time.
- A command completes with an exception status.
- The exception condition is removed before (or upon) the arrival of subsequent in-transit commands.
Example:
- A command terminates with TASK SET FULL (or some other non-ACA error status).
  - The condition may be cleared spontaneously.
  - If so, subsequent commands in-transit may be executed.
Goals
- Feature should be optional
- Prevent a command exception from causing out-of-order command execution,
- Mechanism should be simple.
  - Don't try to optimize error cases.
- Mechanism should be compatible with existing SCSI-2 device drivers,
  - Provided driver is written to be independent of the interconnect (as for CAM).
- Recovery from all 'non ACA' exceptions should be hidden in 'SIM/XPORT' layer and for compatibility with SCSI-2 behavior (as seen by the device driver).
- Device driver recovery from Contingent Allegiance is compatible with SCSI-2.
First Proposal

- Create **optional** new interlock similar to ACA, which is set whenever one of the following statuses is returned:
  - ACA ACTIVE
  - TASK SET FULL
  - RESERVATION CONFLICT
  - BUSY
  - CHECK CONDITION (NACA clear)

- When an exception condition occurs the LUN:
  - Blocks further commands from entering the task set from faulted initiator. Returns special status for such commands.
  - For tasks already in the task set -- Blocks the return of further statuses from the logical unit to faulted initiator.
    - Needed for RESERVATION CONFLICT.
  - The condition must be cleared by the faulted initiator (via a new task management request).
First Proposal
(cont)

- When the initiator's protocol layer detects the exception, its SIM/XPORT layer:
  - Stops sending commands to the logical unit.
  - Automatically flushes all in-transit commands.
  - Marks all flushed commands for resend.
  - Passes the exception status to the application client.
  - Performs recovery based on exception type.
What layer should contain the interlock?

- Protocol Layer
  - Interlock controlled in a protocol-specific manner.
  - Advantage: Possible to optimize based on protocol characteristics.
  - Downside: Different implementation for each interconnect.

- Application layer (Device server)
  - Interlock controlled by standard task management functions.
  - Advantage: One implementation fits any interconnect.
  - Downside: May not be as efficient as protocol layer control.
• Assumptions
  - Host I/O architecture similar to CAM.
  - Initiator's SIM/XPORT layer (or equivalent) retains lists of pending commands in the order sent by the device driver.
    - List of commands waiting to be sent.
    - List of commands that have been sent.
  - A command is removed from the appropriate list:
    - When status is received,
    - When the command is aborted by the application/client.
The initiator’s SIM/XPORT layer, on receiving the exception status:
- Passes the exception status to the application client.
- Stops sending commands to the faulted LUN.
- Flushes in-transit commands.
  - Procedure is protocol-specific.
- Marks flushed commands for resend.
- Clears exception interlock.
- Resends marked commands when requested by application client.

---

**Command Fault Recovery**

![Diagram showing command fault recovery process involving Initiator, SIM/XPORT layer, Task Management Requests and Commands, Status, SCSI Protocol Layer, Logical Unit, Task Set, Interlock, and Target.]

---

**Table: Command Fault Recovery**

<table>
<thead>
<tr>
<th>Initiator</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM/XPORT Layer</td>
<td>SCSI Protocol Layer</td>
</tr>
<tr>
<td>Device Driver</td>
<td>Logical Unit</td>
</tr>
</tbody>
</table>

---

**Diagram Notes:**

- Initiator sends requests to the Target through the SIM/XPORT layer.
- Status is updated and communicated between Initiator and Target.
- Task Management Requests and Commands are handled by the SCSI Protocol Layer.
- Interlock mechanisms ensure data integrity during command fault recovery.
What a CAM-type Device Driver sees:
- Behavior is identical to parallel SCSI.
- Command Queue frozen as for CAM.
  - No out-of-order command execution.
- Contingent Allegiance
  - Sense data automatically preserved.
- Interface for manipulating the queue of unsent SCSI commands is implementation-specific but protocol independent.

What the Device Driver doesn't see:
- Management of 'pending command list' by SIM/XPORT layer.
The initiator’s SIM/XPORT layer, on receiving the exception status:
- Passes the exception status to the device driver.
- Stops sending commands to the faulted LUN.
- **Does not** flush in-transit commands. (This is done by the device driver using the ACA interlock).
- Passes all completion statuses to device driver.
- No flushed commands to resend.
- Application client may send one or more ACA commands followed by CLEAR ACA request.
Status of proposal:
- WG Observed that proposal was similar to ACA. Therefore ACA should be extended.
- 96-173R1 written to extend ACA mechanism to handle all exceptions.
- Glitches found by WG
  - Proposal broke the following ACA properties:
    - ACA can only be in effect for one initiator at a time.
    - ACAs don't nest. (What happens when a command terminates with an ACA ACTIVE status if an ACA is already in effect for that initiator?)

Proposal:
- Revert back to initial approach (new optional interlock).
- Get WG feedback.
- Prepare strawman and post on reflector
- Discuss at next WG.
- Vote up or down.