                Small Computer Systems Interface (SCSI)
                 Command Ordering Considerations with iSCSI

Status of this Memo

Copyright Notice

Abstract

   Internet Small Computer Systems Interface (iSCSI) is a Small Computer
   Systems Interface (SCSI) transport protocol designed to run on top of
   TCP. The iSCSI session abstraction is equivalent to the classic SCSI
   "I_T nexus", which represents the logical relationship between an
   Initiator and a Target (I and T) required in order to communicate via
   the SCSI family of protocols. The iSCSI session provides an ordered
   command delivery from the SCSI initiator to the SCSI target. This
   document goes into the design considerations that led to the iSCSI
   session model as it is defined today, relates the SCSI command
   ordering features defined in T10 specifications to the iSCSI
   concepts, and finally provides guidance to system designers on how
   true command ordering solutions can be built based on iSCSI.

Table of Contents

1. Introduction

   iSCSI is a SCSI transport protocol ([iSCSI]) designed to enable
   running SCSI application protocols on TCP/IP networks, including
   potentially the Internet. Given the size and scope of the Internet,
   iSCSI thus enables some exciting new SCSI applications. Potential
   new application areas for exploiting iSCSI's value include the
   following:

      a) Larger (diameter) Storage Area Networks (SANs) than had been
         possible until now
      b) Asynchronous remote mirroring
      c) Remote tape vaulting

   Each of these applications takes advantage of the practically
   unlimited geographical distance that iSCSI enables between a SCSI
   initiator and a SCSI target. In each of these cases, because of the
   long delays involved, there is a very high incentive for the
   initiator to stream SCSI commands back-to-back without waiting for
   the SCSI status of previous commands. Command streaming may be
   employed primarily by two classes of applications - while one class
   may not particularly care about ordered command execution, the other
   class does rely on ordered command execution (i.e. there is an
   application-level dependency on the ordering among SCSI commands).
   As an example, cases b) and c) listed earlier clearly require ordered
   command execution. A mirroring application does not want the writes
   to be committed out of order on the remote SCSI target, so as to

preserve the transactional integrity of the data on that target. To
summarize, SCSI command streaming, when coupled with the guarantee of
ordered command execution on the SCSI target, is extremely valuable
for a critical class of applications in long-latency networks.

This document reviews the various protocol considerations in
designing storage solutions that employ SCSI command ordering. This
document also analyzes and explains the design intent of [iSCSI] with
respect to command ordering.

2. Definitions and Acronyms

2.1. Definitions

   - I_T nexus: [SAM2] defines the I_T nexus as a relationship between
      a SCSI initiator port and a SCSI target port. [iSCSI] defines an
      iSCSI session as the iSCSI representation of an I_T nexus. In the
      iSCSI context, the I_T nexus (i.e. the iSCSI session) is a
      relationship between an iSCSI initiator's end of the session (SCSI
      Initiator Port) and the iSCSI target's Portal Group (SCSI Target
      Port).

   - PDU (Protocol Data Unit): An iSCSI initiator and iSCSI target
      communicate using iSCSI protocol messages. These messages are
      called "iSCSI protocol data units" (iSCSI PDUs).

   - SCSI device: A SCSI device is an entity that contains one or more
      SCSI ports that are connected to a service delivery subsystem and
      supports SCSI application protocols. In the iSCSI context, the
      SCSI Device is the component within an iSCSI Node that provides
      the SCSI functionality. The SCSI Device Name is defined to be the
      iSCSI Name of the node.

   - Session: A group of logically related iSCSI connections that link
      an initiator with a target form a session (equivalent to a SCSI
      I-T nexus). The number of participating iSCSI connections within
      an iSCSI session may vary over time. The multiplicity of
      connections at the iSCSI level is completely hidden for the SCSI
      layer - each SCSI port in an I_T nexus sees only one peer SCSI
      port across all the connections of a session.

2.2. Acronyms

```
   Acronym                      Definition
   ---------------------------------------------------------------
   ACA                          Auto Contingent Allegiance
   ASC                          Additional Sense Code
   ASCQ                         Additional Sense Code Qualifier
   CRN                          Command Reference Number
   IETF                         Internet Engineering Task Force
   ISID                         Initiator Session Identifier
   ITT                          Initiator Task Tag
   LU                           Logical Unit
   LUN                          Logical Unit Number
   NIC                          Network Interface Card
   PDU                          Protocol Data Unit
   TMF                          Task Management Function
   TSIH                         Target Session Identifying Handle
   SAN                          Storage Area Network
   SCSI                         Small Computer Systems Interface
   TCP                          Transmission Control Protocol
   UA                           Unit Attention
   WG                           Working Group
```

3. Overview of the iSCSI Protocol

3.1. Protocol Mapping Description

   The iSCSI protocol is a mapping of the SCSI remote procedure
   invocation model (see [SAM2]) over the TCP protocol.

   SCSI's notion of a task maps to an iSCSI task. Each iSCSI task is
   uniquely identified within that I_T nexus by a 32-bit unique
   identifier called Initiator Task Tag (ITT). The ITT is both an iSCSI
   identifier of the task and a classic SCSI task tag.

   SCSI commands from the initiator to the target are carried in iSCSI
   requests called SCSI Command PDUs. SCSI status back to the initiator
   is carried in iSCSI responses called SCSI Response PDUs. SCSI Data-
   out from the initiator to the target is carried in SCSI Data-Out
   PDUs, and the SCSI Data-in back to the initiator is carried in SCSI
   Data-in PDUs.

3.2. The I_T Nexus Model

   In the iSCSI model, the SCSI I_T nexus maps directly to the iSCSI
   session, which is an iSCSI protocol abstraction spanning one or more
   TCP connections. The iSCSI protocol defines the semantics in order
   to realize one logical flow of bidirectional communication on the I_T
   nexus, potentially spanning multiple TCP connections (as many as
   2^16). The multiplicity of iSCSI connections is thus completely
   contained at the iSCSI layer, while the SCSI layer is presented with
   a single I_T nexus, even in a multi-connection session. A session
   between a pair of given iSCSI nodes is identified by the session
   identifier (SSID) and each connection within a given session is
   uniquely identified by a connection identifier (CID) in iSCSI. The
   SSID itself has two components - Initiator Session Identifier (ISID)
   and a Target Session Identifying Handler (TSIH) - each identifying
   one end of the same session.

   There are four crucial functional facets of iSCSI that together
   present this single logical flow abstraction to the SCSI layer, even
   with an iSCSI session spanning across multiple iSCSI connections.

      a) Ordered command delivery: A sequence of SCSI commands that is
         striped across all the connections in the session is
         "reordered" by the target iSCSI layer into an identical
         sequence based on a Command Sequence Number (CmdSN) that is
         unique across the session. The goal is to achieve bandwidth
         aggregation from multiple TCP connections, but to still make it
         appear to the target SCSI layer as if all the commands had
         travelled in one flow.

      b) Connection allegiance: All the PDU exchanges for a SCSI
         Command, up to and including the SCSI Response PDU for the
         Command, are required to flow on the same iSCSI connection at
         any given time. This again is intended to hide the multi-
         connection nature of a session because the SCSI layer on either
         side will never see the PDU contents out of order (e.g., status
         cannot bypass read data for an initiator).

      c) Task set management function handling: [iSCSI] specifies an
         ordered sequence of steps for the iSCSI layer on the SCSI
         target in handling the two SCSI task management functions
         (TMFs) that manage SCSI task sets. The two TMFs are ABORT TASK
         SET that aborts all active tasks in a session, and CLEAR TASK
         SET that clears the tasks in the task set. The goal of the
         sequence of steps is to guarantee that the initiator receives
         the SCSI Response PDUs of all unaffected tasks before the TMF
         Response itself arrives, regardless of the number of
         connections in the iSCSI session. This operational model is

again intended to preserve the single flow abstraction to the
SCSI layer.

d) Immediate task management function handling: Even when a TMF
request is marked as "immediate" (i.e. only has a position in
the command stream, but does not consume a CmdSN), [iSCSI]
defines semantics that require the target iSCSI layer to ensure
that the TMF request is executed as if the commands and the TMF
request were all flowing on a single logical channel. This
ensures that the TMF request will act on tasks that it was
meant to manage.

The following sections will analyze the "Ordered command delivery"
aspect in more detail, since command ordering is the focus of this
document.

3.3. Ordered Command Delivery

3.3.1. Questions

A couple of important questions related to iSCSI command ordering
were considered early on in the design of the iSCSI protocol. The
questions were:

a) What should be the command ordering behavior required of iSCSI
implementations in the presence of transport errors, such as
errors that corrupt the data in a fashion that is not detected
by the TCP checksum (e.g., two offsetting bit flips in the same
bit position), but is detected by the iSCSI CRC digest?

b) Should [iSCSI] require both initiators and targets to use
ordered command delivery?

Since the answers to these questions are critical to the
understanding of the ordering behavior required by the iSCSI
protocol, the following sub-sections consider them in more detail.

3.3.2. The Session Guarantee

The final disposition of question a) in section 3.3.1 was reflected
in [RFC3347], "iSCSI MUST specify strictly ordered delivery of SCSI
commands over an iSCSI session between an initiator/target pair, even
in the presence of transport errors." Stated differently, an iSCSI
digest failure, or an iSCSI connection termination, must not cause
the iSCSI layer on a target to allow executing the commands in an
order different from that intended (as indicated by the CmdSN order)
by the initiator. This design choice is enormously helpful in
building storage systems and solutions that can now always assume

   command ordering to be a service characteristic of an iSCSI
   substrate.

   Note that by taking the position that an iSCSI session always
   guarantees command ordering, [iSCSI] was indirectly implying that the
   principal reason for the multi-connection iSCSI session abstraction
   was to allow ordered bandwidth aggregation for an I_T nexus. In
   deployment models where this cross-connection ordering mandated by
   [iSCSI] is deemed expensive, a serious consideration should be given
   to deploying multiple single-connection sessions instead.

3.3.3. Ordering Onus

   The final resolution of b) in section 3.3.1 by the iSCSI protocol
   designers was in favor of not always requiring the initiators to use
   command ordering. This resolution is reflected in dropping the
   mandatory ACA usage requirement on the initiators, and allowing an
   ABORT TASK TMF to plug a command hole etc., since these are conscious
   choices an initiator makes in favor of not using ordered command
   delivery. The net result can be discerned by a careful reader of
   [iSCSI] - the onus of ensuring ordered command delivery is always on
   the iSCSI targets, while the initiators may or may not utilize
   command ordering. iSCSI targets, being the servers in the client-
   server model, do not really attempt to establish whether or not a
   client (initiator) intends to take advantage of command ordering
   service, but instead simply always provide the guaranteed delivery
   service. The rationale here is that there are inherent SCSI and
   application-level dependencies, as we shall see in building a command
   ordered solution, that are beyond the scope of [iSCSI], to mandate or
   even discern the intent with respect to the usage of command
   ordering.

3.3.4. Design Intent

   To summarize the design intent of [iSCSI]:

   The service delivery subsystem (see [SAM2]) abstraction provided by
   an iSCSI session is guaranteed to have the intrinsic property of
   ordered delivery of commands to the target SCSI layer under all
   conditions. Consequently, the guarantee of the ordered command
   delivery is across the entire I_T nexus spanning all the LUs that the
   nexus is authorized to access. It is the initiator's discretion as
   to whether or not this property will be used.

4. The Command Ordering Scenario

   A storage systems designer working with SCSI and iSCSI has to
   consider the following protocol features in SCSI and iSCSI layers,
   each of which has a role to play in realizing the command ordering
   goal.

4.1. SCSI Layer

   The SCSI application layer has several tools to enforce ordering.

4.1.1. Command Reference Number (CRN)

   CRN is an ordered sequence number which, when enabled for a device
   server, increments by one for each I_T_L nexus (see [SAM2]). The one
   notable drawback with CRN is that there is no SCSI-generic way (such
   as through mode pages) to enable or disable the CRN feature. [SAM2]
   also leaves the usage semantics of CRN for the SCSI transport
   protocol, such as iSCSI, to specify. [iSCSI] chose not to support
   the CRN feature for various reasons.

4.1.2. Task Attributes

   [SAM2] defines the following four task attributes - SIMPLE, ORDERED,
   HEAD OF QUEUE, and ACA. Each task to an LU may be assigned an
   attribute. [SAM2] defines the ordering constraints that each of
   these attributes conveys to the device server that is servicing the
   task. In particular, judicious use of ORDERED and SIMPLE attributes
   applied to a stream of pipelined commands could convey the precise
   execution schema for the commands that the initiator issues, provided
   the commands are received in the same order on the target.

4.1.3. Auto Contingent Allegiance (ACA)

   ACA is an LU-level condition that is triggered when a command (with
   the NACA bit set to 1) completes with CHECK CONDITION. When ACA is
   triggered, it prevents all commands other than those with the ACA
   attribute from executing until the CLEAR ACA task management function
   is executed, while blocking all the other tasks already in the task
   set. See [SAM2] for the detailed semantics of ACA. Since ACA is
   closely tied to the notion of a task set, one would ideally have to
   select the scope of the task set (by setting the TST bit to 1 in the
   control mode page of the LU) to be per-initiator in order to prevent
   command failures in one I_T_L nexus from impacting other I_T_L
   nexuses through ACA.

4.1.4. UA Interlock

   When UA interlock is enabled, the logical unit does not clear any
   standard Unit Attention condition reported with autosense, and in
   addition, establishes a Unit Attention condition when a task is
   terminated with one of BUSY, TASK SET FULL, or RESERVATION CONFLICT
   statuses. This so-called "interlocked UA" is cleared only when the
   device server executes an explicit REQUEST SENSE ([SPC3]) command
   from the same initiator. From a functionality perspective, the scope
   of UA interlock today is slightly different from ACA's because it
   enforces ordering behavior for completion statuses other than CHECK
   CONDITION, but otherwise conceptually has the same design intent as
   ACA. On the other hand, ACA is somewhat more sophisticated because
   it allows special "cleanup" tasks (ones with ACA attribute) to
   execute when ACA is active. One of the principal reasons UA
   interlock came into being was that SCSI designers wanted a command
   ordering feature without the side effects of using the aforementioned
   TST bit in the control mode page.

4.2. iSCSI Layer

   As noted in section 3.2 and section 3.3, the iSCSI protocol enforces
   and guarantees ordered command delivery per iSCSI session using the
   CmdSN, and this is an attribute of the SCSI transport layer. Note
   further that any command ordering solution that seeks to realize
   ordering from the initiator SCSI layer to the target SCSI layer would
   be of practical value only when the command ordering is guaranteed by
   the SCSI transport layer. In other words, the related SCSI
   application layer protocol features such as ACA etc. are based on the
   premise of an ordered SCSI transport. Thus, iSCSI's command ordering
   is the last piece in completing the puzzle of building solutions that
   rely on ordered command execution, by providing the crucial guarantee
   that all the commands handed to the initiator iSCSI layer will be
   transported and handed to the target SCSI layer in the same order.

5. Connection Failure Considerations

   [iSCSI] mandates that when an iSCSI connection fails, the active
   tasks on that connection must be terminated if not recovered within a
   certain negotiated time limit. When an iSCSI target does terminate
   some subset of tasks due to iSCSI connection dynamics, there is a
   danger that the SCSI layer would simply move on to the next tasks
   waiting to be processed and execute them out-of-order unbeknownst to
   the initiator SCSI layer. To preclude this danger, [iSCSI] further
   mandates the following:

    a) The tasks terminated due to the connection failure must be
       internally terminated by the iSCSI target "as if" due to a
       CHECK CONDITION. While this particular completion status is
       never communicated back to the initiator, the "as if" is still
       meaningful and required because if the initiator were using ACA
       as the command ordering mechanism of choice, a SCSI-level ACA
       will be triggered due to this mandatory CHECK CONDITION. This
       addresses the aforementioned danger.

    b) After the tasks are terminated due to the connection failure,
       the iSCSI target must report a Unit Attention condition on the
       next command processed on any connection for each affected
       I_T_L nexus of that session. This is required because if the
       initiator were using UA interlock as the command ordering
       mechanism of choice, a SCSI-level UA will trigger a UA-
       interlock. This again addresses the aforementioned danger.
       iSCSI targets must report this UA with the status of CHECK
       CONDITION, and the ASC/ASCQ value of 47h/7Fh ("SOME COMMANDS
       CLEARED BY ISCSI PROTOCOL EVENT").

6. Command Ordering System Considerations

   In general, command ordering is automatically enforced if targets and
   initiators comply with the iSCSI specification. However, listed
   below are certain additional related implementation considerations
   for the iSCSI initiators and targets to take note of.

    a) Even when all iSCSI and SCSI command ordering considerations
       earlier noted in this document were applied, it is beneficial
       for iSCSI initiators to proactively avoid scenarios that would
       otherwise lead to out-of-order command execution. This is
       simply because the SCSI command ordering features such as UA
       interlock are likely to be costlier in performance when they
       are allowed to be triggered. [iSCSI] provides enough guidance
       on how to implement this proactive detection of PDU ordering
       errors.

    b) The whole notion of command streaming does of course assume
       that the target in question supports command queueing. An
       iSCSI target desirous of supporting command ordering solutions
       should ensure that the SCSI layer on the target supports
       command queuing. The remote backup (tape vaulting)
       applications that iSCSI enables make an especially compelling
       case that tape devices should give a very serious consideration
       to supporting command queuing, at least when used in
       conjunction with iSCSI.

   c) An iSCSI target desirous of supporting high-performance command
      ordering solutions that involve specifying a description of
      execution schema should ensure that the SCSI layer on the
      target in fact does support the ORDERED and SIMPLE task
      attributes.

   d) There is some consideration of expanding the scope of UA
      interlock to encompass CHECK CONDITION status, and thus make it
      the only required command ordering functionality of
      implementations to build command ordering solutions. Until
      this is resolved in T10, the currently defined semantics of UA
      interlock and ACA warrant implementing both features by iSCSI
      targets desirous of supporting command ordering solutions.

7. Reservation Considerations

   [iSCSI] describes a "principle of conservative reuse" that encourages
   iSCSI initiators to reuse the same ISIDs (see section 3.2) to various
   SCSI target ports, in order to present the same SCSI initiator port
   name to those target ports. This is in fact a very crucial
   implementation consideration that must be complied with. [SPC3]
   mandates the SCSI targets to associate persistent reservations and
   the related registrations with the SCSI initiator port names whenever
   they are required by the SCSI transport protocol. Since [iSCSI]
   requires the mandatory SCSI initiator port names based on ISIDs,
   iSCSI targets are required to work off the SCSI initiator port names,
   and thus indirectly the ISIDs, in enforcing the persistent
   reservations.

   This fact has the following implications for the implementations:

   a) If a persistent reservation/registration is intended to be used
      across multiple SCSI ports of a SCSI device, the initiator
      iSCSI implementation must use the same ISID across associated
      iSCSI sessions connecting to different iSCSI target portal
      groups of the SCSI device.

   b) If a persistent reservation/registration is intended to be used
      across the power loss of a SCSI target, the initiator iSCSI
      implementation must use the same ISIDs as before in
      re-establishing the associated iSCSI sessions upon subsequent
      reboot in order to rely on the persist through power loss
      capability.

8. Security Considerations

   For security considerations in using the iSCSI protocol, refer to the
   Security Considerations section in [iSCSI]. This document does not
   introduce any additional security considerations other than those
   already discussed in [iSCSI].

9. References

9.1. Normative References

   [iSCSI]   Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M. and
              E. Zeidner, "Internet Small Computer Systems Inferface
              (iSCSI)", RFC 3720, May 2004.

   [SAM2]   ANSI INCITS.366:2003 SCSI Architecture Model - 2 (SAM-2).

9.2. Informative References

   [RFC793]   Postel, J., "Transmission Control Protocol", STD 7, RFC
              793, September 1981.

   [RFC2119] Bradner, S., "Key Words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC3347] Krueger, M. and R. Haagens, "iSCSI Requirements and Design
              Considerations", RFC 3347, July 2002.

   [SPC3]   INCITS T10/1416-D, SCSI Primary Commands-3 (SPC-3).

10. Acknowledgments

   We are grateful to the IPS working group whose work defined the iSCSI
   protocol. Thanks also to David Black (EMC) who encouraged the
   publication of this document. Special thanks to Randy Haagens (HP)
   for his insights on the topic of command ordering. Thanks are also
   due to Elizabeth Rodriguez for carefully reviewing this document.

11. Authors' Addresses

   Mallikarjun Chadalapaka
   Hewlett-Packard Company
   8000 Foothills Blvd.
   Roseville, CA 95747-5668, USA

   Phone: +1.916.785.5621
   EMail: cbm@rose.hp.com

   Rob Elliott
   Hewlett-Packard Company
   MC140801
   PO Box 692000
   Houston, TX 77269-2000 USA

   Phone: +1.281.518.5037
   EMail: elliott@hp.com

12. Full Copyright Statement

Intellectual Property

Acknowledgement