Introduction

- The linux SCSI stack has a long-standing history
- Including an error recovery strategy
- Has been in the linux kernel since time immemorial
- And what with it being heavily used, it will have been tested thoroughly by now.
Introduction

- The linux SCSI stack has a long-standing history
- Including an error recovery strategy
- Has been in the linux kernel since time immemorial
- And what with it being heavily used, it will have been tested thoroughly by now.

... Or so one would hope
Introduction

• The linux SCSI stack has a long-standing history
• Including an error recovery strategy
• Has been in the linux kernel since time immemorial
• And what with it being heavily used, it will have been tested thoroughly by now.
  … Or so one would hope
  … And then real life kicked in
An angry customer

• Received a customer call:

“One of my system took more than two hours to recover from a SCSI error, despite multipath being active and all other paths had been ok. During that time no I/O had been possible. Isn't multipath supposed to handle these situations?”
An angry customer

• Received a customer call:

“One of my system took more than two hours to recover from a SCSI error, despite multipath being active and all other paths had been ok. During that time no I/O had been possible. Isn't multipath supposed to handle these situations?”

… Good question. So what happened here?
SCSI Error Handling
SCSI Error handling in general

• SCSI is governed by T-10 standards
• Everything regarding SCSI commands and SCSI command handling is specified:
  - SCSI command specifications (SPC, SBC, etc)
  - SCSI command transport (SAS, FC, iSCSI etc)
  - SCSI architecture model (SAM)
SCSI error recovery

• Some hints can be glanced from the SCSI architectural model

• Defines *Task Management Functions* to control commands and command sets:
  - Task abort
  - Task set abort
  - LUN Reset

• But error recovery itself is not specified
SCSI error recovery implementations

- No specification, so devise your own
- Implementation based on architecture details, with tweaks accumulating over time
SCSI-EH on Linux
Linux SCSI EH

- Originally implemented in Linux 2.2, based on the then-up-to-date SCSI parallel HBAs
- Improvement over the prior, simple, error recovery procedures
- Modelled around the principles of parallel SCSI:
  - Bus topology
  - Bus is being driven by the HBA
  - Transaction between a single initiator and single targets only
  - Bus is capable of handling a single transaction at a time
SCSI Parallel bus topology
EH Principles

- Retry the command
- Quiesce bus prior to start EH
- Invoke EH strategy for each device referred to by a failed command
- Escalate to higher EH levels on failure
- Verify device operation after successful completion of EH strategy routine
EH Recovery Strategy

• Command abort
• Send Test Unit Ready
• LUN Reset
• Target Reset
• Bus Reset
• HBA Reset
• Offline device
EH recovery strategy

Command timeout

abort cmd

Command error

LUN Reset

Target Reset

Bus Reset

Host Reset

Send TUR

Device offline
EH Recovery workflow

• Each failed command will be added to a list of failed commands
• Process this list after quiesce has been reached
• Each failed command is subjected to the error escalation strategy
• A command is considered \textit{recovered} once an error recovery routine succeeds
EH Recovery cleanup

• A successful recovery is not identical with a working device:
  - A successful LUN RESET just means we've been able to send a LUN RESET command, **NOT** that the device actually has been reset
  - Nor does it mean that the reset was able to fix the original issue

• Verify the recovery

• Send TEST UNIT READY command to verify the device is working
SCSI EH on FibreChannel
FibreChannel topology

• On FibreChannel (FC) the bus is no longer controlled by the HBA

• HBA participates on a shared network, which has an independent lifetime than the HBA

• SCSI devices (remote FC ports) are independent on the HBA

• Connection between the HBA and the remote ports might drop at any time (I_T nexus loss)
FC topology
FC and multipathing

- Multipath has been implemented to avoid temporary I/O failure
- Connect a single device via several paths to provide enhanced reliability
- Any I_T Nexus loss would translate into an I/O error, invoking SCSI EH
- SCSI EH would stop I/O etc.
- Multipath would stop until SCSI EH is finished
I_T nexus loss and SCSI EH

- Lower EH escalation steps require working communication with the device
- For an I_T Nexus loss this communication doesn't work, causing EH failure for those steps
- SCSI EH would cause a host reset, and offline the device after that
- Path cannot be recovered.
fc_block_scsi_eh() and dev_loss_tmo

- **fc_block_scsi_eh()**: Avoid any I_T Nexus Loss induced error by checking the connection state prior to call any EH recovery routine, waiting for the connection state to stabilize.

- **FAST_IO_FAIL**: Add a flag to the request to avoid any retry in case of I_T Nexus loss failure.

- **dev_loss_tmo**: Add a timer tracking I_T Nexus loss; once the timer expires the remote port is assumed to be gone and will be deleted from the system.
'Improved' EH for FibreChannel

• FAST_IO_FAIL flag suppresses command retries
• Distinct error code 'DID_TRANSPORT_DISRUPTED' to be returned in case of I_T Nexus loss
• Short-circuit SCSI EH by prefixing each EH routine with fc_block_scsi_eh()
  → Side-step EH for FibreChannel
SCSI EH on libata
Libata implementation

- Re-implement S-ATA support on top of SCSI
- Successor of the older IDE stack
- S-ATA error handling very rudimentary: commands either succeed or run into a timeout.
- Standard SCSI EH doesn't work, as the EH recovery routines have no equivalent on S-ATA
- Implement different EH routine via .eh_strategy_handler
SCSI EH on SAS
SAS and SCSI EH

- Working well with stock SCSI EH
SAS and SCSI EH

- Working well with stock SCSI EH
- Until someone connected a S-ATA CD-ROM
SAS and SCSI EH

- Working well with stock SCSI EH
- Until someone connected a S-ATA CD-ROM
- Suddenly the entire system stalled every 5 seconds
  … WTF?
Libata oddities

- Libata has a 1:1 topology: one SCSI device maps to one SCSI host.
- The libata error recovery stops the SCSI host, figures out what's wrong with sending various commands, retrains the link etc until the device respond again.
- Sadly, a CD-ROM with empty slot will cause an ATA error as there's no medium present.
- And the linux kernel implement CD-ROM polling within the kernel.
SAS topology
SAS and libata

- SAS HBAs offload S-ATA devices to libata stack
- S-ATA devices show up alongside normal SAS devices as a 'normal' LUN.
- Each SAS HBA will be represented by a single SCSI host
Mixed SAS/S-ATA topology

libsas

libata
CD-ROM polling on SAS/libata

• Kernel polls CD-ROM
CD-ROM polling on SAS/libata

- Kernel polls CD-ROM
- CD-ROM on libata registers an error as CD-ROM medium is not present.
CD-ROM polling on SAS/libata

- Kernel polls CD-ROM
- CD-ROM on libata registers an error as CD-ROM medium is not present.
- Libata error recover starts.
CD-ROM polling on SAS/libata

- Kernel polls CD-ROM
- CD-ROM on libata registers an error as CD-ROM medium is not present.
- Libata error recover starts.
- SCSI Host is stopped.
CD-ROM polling on SAS/libata

- Kernel polls CD-ROM
- CD-ROM on libata registers an error as CD-ROM medium is not present.
- Libata error recover starts.
- SCSI Host is stopped.
- All I/O to LUNs connected to that Host is stopped.
CD-ROM polling on SAS/libata

- Kernel polls CD-ROM
- CD-ROM on libata registers an error as CD-ROM medium is not present.
- Libata error recover starts.
- SCSI Host is stopped.
- All I/O to LUNs connected to that Host is stopped.
- For a single SAS HBA: the entire I/O will be stopped.
CD-ROM polling on SAS/libata

- Kernel polls CD-ROM
- CD-ROM on libata registers an error as CD-ROM medium is not present.
- Libata error recover starts.
- SCSI Host is stopped.
- All I/O to LUNs connected to that Host is stopped.
- For a single SAS HBA: the entire I/O will be stopped.

Oops …
SAS EH modification

- Not use standard SCSI EH routines
- Implement separate .eh_strategy_handler for SAS
So where do we stand now?
Analysis of the customer problem

• Switch firmware issues caused the HBA to **not** detect a remote port failure
• HBA continues to send I/O to the removed rport
  • *(wait 5 x 30 seconds)*
• First I/O times out
• SCSI EH starts, waiting for all outstanding commands
  • *(wait for another 5 x 30 seconds)*
• SCSI EH recovery starts after the **last** command timed out
Analysis of the customer problem

• EH recovery, first level: command abort
  - send command abort for the first command
  - *(wait for timeout)*
  - Abort the command abort
  - *(continue for all commands)*

• Escalate to next level: LUN reset
Analysis of the customer problem

- EH recovery, second level: LUN reset
  - send LUN Reset for the first device
  - *(wait for timeout)*
  - *(continue for all devices)*

- Escalate to next level: Target reset
Analysis of the customer problem

• EH recovery, third level: Target reset
  - send Target Reset for the first device
  - (wait for timeout)
  - (continue for all targets)

• Escalate to next level: Bus reset

• → Target Reset is deprecated with SPC-3
Analysis of the customer problem

- EH recovery, third level: Bus reset
  - FC does not have the concept of a 'bus', so most HBAs emulate 'Bus reset' by sending 'Target Reset' to all attached ports
  - (wait for timeout)
  - (continue for all targets)

- Escalate to next level: Host reset
Analysis of the customer problem

• EH recovery, forth level: Host reset
  - Issue Host reset
  - Host reset re-scans the attached remote ports
  - Remote port status in sync again
• EH recovery success
• Send TEST UNIT READY to all devices
• EH finished
SCSI EH Redesign
Current SCSI EH usage

- FC: Side-step SCSI EH
- Libata: separate EH handler
- SAS: separate EH handler
- Only parallel SCSI and iSCSI are still using stock SCSI EH
- Maybe we should be updating SCSI EH to make it more useful …
SCSI EH Redesign

• Overall goals:
  ‒ Inline command aborts
  ‒ Limit overall SCSI EH runtime
  ‒ Release commands as early as possible
  ‒ Reduce cross-speak during higher EH levels
  ‒ Check for I_T Nexus loss
Inline command aborts

- Command timeouts can occur on FC with faulty SFPs
- Command abort has no dependency on other commands, just the originating command
- Send command abort once the timeout triggers, without waiting for EH to start
- Patchset posted to linux-scsi
- Reduce SCSI EH turn-around time by half (!)
Limit overall EH runtime

- Currently EH runtime is unbounded
- Hard to define system timeout, eg in cluster environment
- Implement an 'eh_deadline' setting
- After eh_deadline is reached SCSI EH drops down to host reset
- Patchset posted to linux-scsi
Release commands early

- SCSI EH keeps failed command in a list
- Commands will be completed after EH is finished
- Multipath failover can only happen after the command has been completed
- After LUN Reset all commands are discarded
- But: LUN Reset might fail, leaving commands in an unclear state (terminated? Not terminated?)
Reduce cross-speak at higher levels

- LUN Reset will terminate all I/O on that LUN, regardless on the initiator
- Spurious command aborts in multipath or cluster scenario
- Split 'LUN Reset' in two different stages:
  - Use 'Task Set abort' to terminate outstanding I/O
  - Use 'LUN Reset' to actually reset the LUN
- Remove Target Reset, deprecated
- Do not implement 'bus reset' on FC
Check for I_T Nexus loss

- On FibreChannel SCSI EH cannot work during I_T Nexus Loss
- Current workaround is to wait in SCSI EH until dev_loss_tmo/fast_io_fail_tmo put the remote port into a definite state
- Implement an I_T Nexus reset EH step which is responsible for resetting the remote port
Proposed SCSI EH strategy

- Send command aborts after timeout
- EH Recovery starts:
  - Block I/O to the device
    - Issue 'Task Set Abort'
  - Block I/O to the target
    - Issue I_T Nexus Reset
    - Complete outstanding command on success
  - Engage current EH strategy
    - LUN Reset, Target Reset etc
EH recovery strategy

Command timeout

abort cmd

Task Set abort

Transport Reset

LUN Reset

Target Reset

Bus Reset

Host Reset

Send TUR

Device offline
SCSI EH discussion points
Early command completion

• Complete commands after 'Abort Task Set'
• Unclear status if 'Abort Task Set' failed
• Easy way:
  ‒ Require LLDDs to not refer to outstanding commands after 'Abort Task Set'
  ‒ But then 'Abort Task Set' cannot really fail, as this is the precise meaning of that function
• Complicated way:
  ‒ Keep the list of commands until one recovery step succeeds
• Best way still to be discussed
Check recovery level status

- Each recovery level can succeed or fail
- 'Success' currently only means that the recovery step has executed
- It does not mean that the recovery step did anything to correct the situation
- Separate verification required
- Action depends on the recovery level
Most recent sources are available at

git://github.com/hreinecke/scsi-devel

Thank you.
Unpublished Work of SUSE. All Rights Reserved.
This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer
This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.