



LSI LOGIC®

LSI Storage HBA Driver Development

Open Source Driver “Lessons Learned” from Dev Mgr perspective

Steve Pope – LSI SCG SW Mgr
July 25th, 2006

25
YEARS

Background:

LSI SCG Storage HBA Linux Driver Development

- LSI Storage Components Group provides SAS/SATA/FC/SCSI Controllers (chips & boards) to OEMs and the Channel
- Goal for today: tell a story from an LSI development manager perspective
- Fall of 2005, Eric Moore began working closer with Linux Community
- Eric Moore is LSI Fusion MPT Linux Driver Maintainer
 - 2.4 Kernel – SLES8, RH3
 - 2.6 Kernel – SLES9, RH4
 - 2.6.16 - SLES10 (3.03.10+), 2.6.18 RH5 new API (3.04.01)
 - Story: Need to merge SLES10 sp1 and RH5 on 3.04.xx path

Lessons Learned:

Driver Changes made in response to Community Pressures

- 2.6K Split Drivers (SCSI, SAS, FC)
 - LSI SCG previously used common driver
 - Kernel.org wanted unique drivers
 - Separated by Transport Layer type
- 2.6.16K new SAS Transport Layer
 - Pushed up topology tracking (ports/phys), hot plug hooks, etc.
 - Error handling is still maintained in SAS Driver
 - Story: SAS Transport kept breaking – Eric worked with community
 - Person who complains usually needs to also provide solution
- Added functionality in the SCSI Transport Layer
 - SCSI Domain Validation moved up to Transport Layer
 - Initial proposed solution was incompatible with LSI Solution
 - Further changes had to be made to DV in SCSI Transport Layer
- Challenges of keeping up with Transport Layer
 - API can change any time and is not always backwards compatible
 - Usually keep API for life of given OS (e.g. SLES 9)
 - Pros / Cons to not enforcing backwards compatibility
 - Supporting legacy can hamper progress / performance

Pros & Cons:

Pros of Open Source Driver Development

- + Higher testing exposure via community
- + Can be quickly checked across many developer systems if needed
- + Motivates the person that cares the most about their solution to test, debug, and propose source solution
- + Some customers can develop and test driver code for their niche if you don't have time and resources
- + Source Patches usually submitted by individual who finds the issue.
 - Although it still takes time to interact and approve patches
- + Over time, as more functionality is incorporated in Transport Layer, will be able to thin driver, though you lose some control in the process
- + Ability to get more custom solutions with minimal in house technical resources required
 - ✓ Opens more doors for our Chips / HBAs
 - Story: OEM Example in FC
- + Many OEMs have key technical resources involved in the community, which enables direct access to OEM technical points of contact. This can be overwhelming for key developers though.

Pros & Cons (continued) :

Cons of Open Source Driver Development

- Requires more developers upfront to manage interaction with community
- Vendor unique and OEM unique implementations normally rejected from kernel.org
- Competitors may work to implement changes in the areas outside your component
- Community does not have high motivation to maintain backwards compatibility.
- IHVs have higher development burden in order to provide solution to market without corresponding revenue increase,
 - ✓ **But if you don't play, then you don't play in the market!!**
- Often difficult to determine schedules for changes going into kernel.org and confidence if they will make an OS cutoff or for that matter get killed completely
- ± If not engaged in community activity, then changes may appear to be done in a vacuum – without soliciting your input
- Story: IOCTLs.
- Story: SSP Passthrough

Strategies:

Strategies to work with Open Source Community

- Fully engage your technical lead developer in the Community
- Requires technical prowess. Must be respected in community.
 - Both individual and company.
 - Provide HW samples to community
 - Must be response to e-mails / msg groups on key topics
- SW Architects must be in the loop regarding direction of your Open Source Driver and surrounding components
 - If your development is done in a community vacuum, your solution risks rise steadily with time
- Partner with OEM and Community to discuss ideas whenever possible
- Be willing to modify your approach to be vendor / OEM neutral in implementation
 - May be harder to create Value Add for some IHVs
- Business Case justifications may be required to make changes to your in-box Drivers, beyond what kernel.org has accepted
- Test Strategies
 - Support and work closely with OSDL to increase automated testing