Proceedings of the Linux Symposium

Volume Two

July 19th–22nd, 2006
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc.*
C. Craig Ross, *Linux Symposium*

Review Committee

Jeff Garzik, *Red Hat Software*
Gerrit Huizenga, *IBM*
Dave Jones, *Red Hat Software*
Ben LaHaise, *Intel Corporation*
Matt Mackall, *Selenic Consulting*
Patrick Mochel, *Intel Corporation*
C. Craig Ross, *Linux Symposium*
Andrew Hutton, *Steamballoon, Inc.*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*
David M. Fellows, *Fellows and Carr, Inc.*
Kyle McMartin

Catalyzing Hardware Driver Development

A Case Study in Four Acts

Darrick J. Wong
IBM Linux Technology Center
djwong@us.ibm.com

Alexis Bruemmer
IBM Linux Technology Center
alexisb@us.ibm.com

Don Fry
IBM Linux Technology Center
brazilnut@us.ibm.com

Mark Salyzyn
Adaptec, Inc.
mark_salyzyn@adaptec.com

Abstract

Hardware driver support is perhaps one of the most difficult hurdles to overcome in the march towards world domination. Unfortunately, getting that support for Linux® is not always a trivial task. This process involves, at a minimum, coordination between Linus Torvald's patch lieutenants and the hardware vendors' engineering and legal departments; on a more practical level, the hardware vendors need to build and maintain good communication channels with the various distributions to gather feedback and to solve problems. We participate in those complex interactions to catalyze the development process, and build open source alternatives when that fails. For this presentation, we offer four examples of this catalytic process, and discuss the task of helping hardware vendors to merge functionality, when possible, from an internal driver release train into mainline.

In the first case, we discuss how ongoing maintenance and enhancement work with the longtime mainline resident `pcnet32` driver progresses without much hand-holding from AMD®, the original hardware vendor; the case details the traditional process of collaborative driver development among many enthusiasts. The second case explores working with Adaptec® to evaluate, diagnose, and resolve performance issues with the `aacraid` driver and ongoing work to clean up widespread confusion in comparing versions of the driver. With regards to the third case, we discuss assistance given to Adaptec to address issues blocking the `aic94xx` Serial Attached SCSI driver from entering mainline. In the final case, we discuss negotiating with Adaptec for hardware behavior specifications, implementing a HostRAID® plugin for `dmraid` (presented at OLS 2005), altering device-mapper to make grub work better, and assisting the distributions to support installation and booting off of a `dmraid` device.

1 Introduction

This is not your typical OLS paper. Unlike most papers, which focus on some part of the kernel and describe technical details of work surrounding that component to invite discussion, this paper instead looks at four differ-

ent kernel drivers, emphasizing the issues encountered and resolved as a part of improving driver support in the core kernel and related support packages with the goal of making everything work out of the box. Of particular interest are the resolutions to the questions raised during OSDL®'s Open Driver fora at both LinuxWorld® Expositions in 2005.[1] The first case presented in this paper reflects the classical method through which drivers are developed for Linux and the remaining three involve complex coordinations between the IBM® Linux Technology Center (LTC), hardware vendors, and the Linux community.

2 Working on a Driver Independently (pcnet32)

Don Fry has worked on the pcnet32 driver at IBM for several years, during which time he has fixed a lot of bugs and helped out with pcnet32-related Xen® emulation. In this first case, Mr. Fry's work on the driver is showcased as an example of the classical Linux driver development process.

The pcnet32 driver has been around for a long time. It was written by Thomas Bogendoerfer and is a good driver. It supports many different versions of the chipset manufactured by AMD and sold on many different boards and platforms, as well as both PCI® and VLB variants of the 32-bit successor to the 16-bit Lance. The latest versions of the pcnet32 card also support 64-bit addressing, though that is not supported by the Linux driver.

2.1 Startup and Early Mistakes

In late 2003, Mr. Fry was asked by his employer, IBM, to "harden" the pcnet32 driver, a generic task with as many meanings as there are implementers. Several internal bugs had been filed against pcnet32 and were used as the starting point. Many of the bugs were related to performance problems or hangs, so netperf was used to try to reproduce the bugs. Mr. Fry spent time reading through the driver to understand how it was written and also downloaded the specs for the chipsets from the AMD Web site.

Several bugs were found and a patch was created and submitted to correct them. Unfortunately, Mr. Fry was fairly new to the open source model and rolled several different fixes into one incohesive patch, a common mistake made by people who are not familiar with Linux development techniques. The patches were sent to the maintainer and to the netdev mailing list, but repeated requests for comment over a period of several months were ignored by the maintainer. In early February, it was pointed out that the pcnet32 patch was really several patches that should be broken into functional pieces and resubmitted, after which they were finally accepted. During mid to late February 2004, there was a flurry of patches submitted and accepted that fixed a bunch of little things that had been found internally, as well as other bugs found and fixed by people outside of IBM.

Enhancements to aid debugging and hotplug add/remove support were added by other people at IBM. Mr. Fry sent some requests to AMD to learn where to get hardware to enhance his testing pool and met with little success; it became clear that any driver maintenance work would likely be done without AMD being an active participant. By early April, the first bugs in the newly added code were being fixed and some performance changes were being added to the drivers. Other people inside IBM also contributed code to aid debugging of the driver.

In late April, the changes to pcnet32 were

[1] http://developer.osdl.org/dev/opendrivers/wiki/index.php/Roadmap

being seen and improved upon by others in the community. One of the problems with the `pcnet32` driver is that the interrupts used by the driver had to start at 2 and there was code added long ago in other portions of the kernel to work around this "feature." A patch was submitted by somebody outside of IBM to hack around this conundrum, though Mr. Fry rewrote the patch to eliminate the original problem altogether. It is a common occurrence for somebody to submit a patch for inclusion only to find that someone else disagrees with the approach strongly enough to rework the patch or write their own version; barring flame wars, this is a practice that is strongly encouraged.

This development and further bug fixing were directed at the 2.6 version of Linux, but the same patches were backported to 2.4 since the same bugs existed there and more people were using 2.4 at the time. Some of the changes were to handle rarely taken error cases better, or to use newer helper routines in the kernel, eliminating redundant code in the driver. In any case, continued development of the `pcnet32` driver in both the 2.6 and 2.4 kernels helps to keep the associated hardware relevant and usable into the future, even after many years of service.

Changes were not without mishaps. One change that was not properly tested caused hangs in VMWare®. A fix by a user of the driver on VMWare corrected the problem quickly. With additional testing, the hang and fix were verified on hardware as well. Never submit a patch that has not been tested, even if it only changes one character!

Continued testing revealed more problems that were hit when packet rates increased. If frames were received fast enough, the driver would never exit the receive interrupt routine. Rx Polling Mode (NAPI) was suggested as a better fix, but that has not yet been successfully implemented for this chipset. More discussion will be required among the network stack maintainers to determine the best way to implement this.

One of the problems reported was found by accident when debugging another problem. One of the versions of the chipset behaves differently when some "reserved" bits are not set to zero. The driver did not properly handle reserved bits, which caused all frames to be sent with a "carrier error" even though no error actually occurred. Mr. Fry had been looking for the "carrier error" cause and not making any progress. A comment made regarding a different problem also applied to the "carrier error" case and turned out to be the cause of the malfunction. In this case, it would have been very helpful to have had access to the hardware engineers, because hardware does not always behave as the documentation states.

Sometimes, however, AMD did provide some guidance to fix bugs. One of the algorithms needed to handle the Media Independent Interface was not properly implemented. Working for a large company allowed Mr. Fry to contact the right people to get answers to questions that he had had little success pursuing as an individual.

2.2 People Fix Their Favorite Issues

By working with various people in the Linux community, it has become rather apparent that some people will appear on the mailing lists long enough to fix the features that they think are broken, and promptly disappear. Another version of the chip, the 79C978, had features that a user wanted, so he sent in a patch to enable them. After some suggestions were proposed, Mr. Fry purchased a pair of 79C978 boards to be able to do better regression testing. Other users have submitted patches that allow the driver to resize transmit rings and receive

rings with ethtool, and to work properly with boards that only have fiber-optic interfaces. Indeed, it seems that many of the patches received on the Linux kernel mailing list are one-time events.

2.3 Different Architectures Hit Different Bugs

One benefit of maintaining good relations with people in the Linux community is that they can test code and find bugs in exotic environments to which patch authors might not have access. Specifically, big-endian systems such as POWER® did not display some debugging or error information correctly, and recurring hangs were reported that could not be reproduced at all on i386® systems. By asking debugging questions to the people who had helped out on other problems, Mr. Fry was able to resolve the hangs.

2.4 The Present

In August 2004, Mr. Bogendoerfer sent Mr. Fry a patch to support boards with multiple physical access points (PHYs). He was not ready yet to submit the patch, so Mr. Fry did some testing and made some suggestions. Other `pcnet32` questions from Tony at Allied Telesyn® in October resulted in Mr. Fry obtaining some boards with multiple PHYs and fiber-only interfaces which facilitated debugging later problems. Another user of multi-PHY boards submitted a patch in 2006 to support them. Some e-mail was exchanged in February and March, showing the proposal by Mr. Bogendoerfer. Tony liked Mr. Bogendoerfer's fix better and made some suggestions to improve it. With the increased activity on the multi-PHY front and the additional testing the patches had received, it was time to submit the enhancements; they were accepted in late March 2006.

2.5 The Future

Allied Telesyn has their own version of the `pcnet32` driver that they support because the mainline version does not meet their needs. Perhaps the fix for multiple PHYs will allow them to merge their fork into the new mainline driver because it is painful to maintain multiple versions of a driver and confusing for people with `pcnet32` boards to keep track of the variants of the driver. This forking problem will be explored in the following three case studies. Finally, as bugs continue to be found and fixed, there are features such as NAPI that should be implemented.

3 Co-Development with a Vendor (`aacraid`)

Mark Salyzyn and Alexis Bruemmer worked together to improve the `aacraid` driver that existed in the Red Hat® Enterprise Linux (RHEL) and SUSE Linux Enterprise (SLES) distributions. This second case studies their combined efforts.

The `aacraid` driver supports mid-range Adaptec RAID controllers. It was first included in the 2.4.17 mainline kernel; residing in mainline allowed the driver to stay current. However, the versions of the driver that existed in various distribution kernels were not up-to-date and lacked needed functionality to work effectively with the newest Adaptec RAID controllers. By fall of 2005, there was enough customer demand that an effort began in the IBM LTC to update the `aacraid` driver in both Novell® and Red Hat's Linux distributions.

3.1 Porting from Mainline

In some cases, backporting the mainline kernel driver to a distribution sufficed. As an exam-

ple, for SLES 9, the IBM LTC created a backport patch that was reviewed by Adaptec and then tested on various xSeries® hardware. Performance testing revealed no read/write performance regressions; stress testing also demonstrated that an I/O freeze witnessed with an older version of the driver had been eliminated.

In another case, however, Red Hat wanted an even more advanced version of the driver than what existed in the current mainline. This meant not only backporting code from mainline, but also sorting through a mix of nearly one-hundred patches that were in various stages of the community acceptance process. Specifically, there were patches that were in a queue to be submitted but had not been accepted into either Mark Haverkamp's `aacraid` development tree at Adaptec or downstream into the tree of the SCSI maintainer, James Bottomley; patches that had yet to traverse from Mr. Bottomley's tree to the other gatekeepers such as Andrew Morton and Alan Cox; and patches that had trickled through all the gatekeepers but had yet to be included in Linus' tree. All of these patches were reviewed to determine which would be accepted and which ones Red Hat cared about the most. This enormous process was led by Red Hat's Tom Coughlan and Mark Salyzyn. The final patches for both RHEL 3 and 4 were then tested by all parties involved (Red Hat, Adaptec, IBM LTC) to verify not only that the patch worked, but also that there were no I/O read/write performance degradations. A series of adjustments were made to the original Red Hat patch to increase the I/O read write performance; in some cases an improvement of 40% was witnessed. Lastly, stress testing verified that an I/O freeze had been eliminated.

3.2 Key Involvements by IBM

At this point, the keen reader may wonder why is IBM involved in such situations. In this case, not all of the development needed to be done by IBM, so why are we the middle man? Though IBM does have extensive hardware resources for testing, our role at the IBM LTC goes far beyond providing hardware. Because the IBM LTC deals with hardware vendors, distributions, and the kernel community, our most useful asset is our relationship with these entities. This effort was a great example of this—creating the communication channel between Messrs. Coughlan and Salyzyn allowed Mr. Coughlan to present his concerns about the potential to introduce performance problems, which Mr. Salyzyn could then address. Furthermore, this relationship enabled Adaptec to push for an updated driver in the Red Hat's RHEL releases while addressing Red Hat's regression concerns. Both companies were satisfied with the finished product.

3.3 Future Work for `aacraid`

A tremendous amount of work has been done to improve the `aacraid` driver. However, looking towards the future, there are some issues that need to be addressed with regards to maintaining the code base, as well as processes that need to continue, such as eliminating regressions, and increasing performance.

3.3.1 Version Problems

Regrettably, the version of the `aacraid` driver supported by Adaptec is slightly different from the version that exists in mainline. Because custom patches are often created in order to keep older distribution kernels compatible with

new hardware, the version existing in those kernels varies. Worse yet, the module version numbers are not a reliable comparison factor. For example, the version numbers reported by the Adaptec driver and the RHEL 3 and RHEL 4 drivers are identical (1.1-5[2412]), yet the source code is not. Some coherency between driver levels and version numbers needs to be established.

3.3.2 Performance Verification and Improvement

Extensive performance testing was done on the recently updated `aacraid` driver that exists in RHEL 4 U3. This type of performance testing needs to continue as firmware is updated and new `aacraid` driver updates are released by Adaptec to measure the effect that new driver updates have on performance. Also, it would be interesting to see how Adaptec RAID controllers with the `aacraid` driver stack up to software RAID.

4 Development with Vendor and Community Assistance (`aic94xx`)

Alexis Bruemmer is a member of the IBM LTC team that pushed for the acceptance of the `aic94xx` driver into mainline. This chapter discusses that process.

The Adaptec 94xx series chip is a Serial Attached SCSI disk controller. Currently, there is no driver support for the 94xx controller in mainline or in any distributions. There is an Adaptec-supported open source driver available (the `adp94xx` driver) but, because of its structure, there is no hope of seeing it accepted upstream. Because the 94xx controller is needed to boot the system in many hardware configurations, installing on such a system requires a driver update disk (DUD). The lack of upstream solutions is a problem for anyone using 94xx controllers. Adaptec, aware of the importance of having an in-box Linux-based solution for customers using their controller, began development on a driver that was closer to the Serial Attached SCSI driver design that the kernel community desired. In early 2005, Adaptec posted the `aic94xx` driver, an open source, mainline friendly, 94xx controller driver written by Luben Tuikov.[2] Unfortunately, there were two main issues with this driver: much of the code was inadequately tested and did not always work correctly, and Mr. Tuikov opted to create an entirely new Serial Attached SCSI transport layer, `sas_class`, instead of building off the existing transport layer, `scsi_transport_sas`. These issues would have to be resolved before the driver could be accepted upstream.

4.1 Uncovering the Bugs

A small team made up of both Adaptec and IBM LTC members worked together to test the `aic94xx` driver on as many hardware platforms as possible to try to uncover and resolve bugs. A list of approximately ten issues were identified, prioritized, and assigned accordingly. These bugs ranged from incompatibility on certain architectures to race conditions during boot up. Unfortunately, it seemed that once one bug was resolved, two more were uncovered. For example, when a working solution was found for the race condition encounter during boot up, the team was finally able to boot a machine with a Serial Attached SCSI expander, only to find that the expander code was highly unreliable. Just as the IBM LTC team began to diagnose and address problems with the expander code, Mr. Bottomley decided to begin

[2] Now maintained by Rob Tarte.

the Serial Attached SCSI transport layer merge process that will be explained in the next section. In the process of this merge, the expander code and the discovery code changed more dramatically than the team predicted, making all previous boot-up and expander patches obsolete. Even with the many setbacks and project re-directions, however, progress is slowly being made on each bug, and the `aic94xx` driver is stabilizing as time goes on.

4.2 The Great Merge

Besides providing fixes for the existing bugs in the `aic94xx` driver, a merge between the `sas_class` and `scsi_transport_sas` transport layers needed to be performed in order to eliminate redundant code. This daunting task was tackled by Mr. Bottomley. The IBM LTC assisted him by providing hardware, aiding in the test process, as well as proposing solutions for bugs in the merged code. Through the collaborative efforts of Mr. Bottomley and the IBM LTC, there has been a successful merge between the `sas_class` (now named `scsi_transport_sas_domain`) and the `scsi_transport_sas` layers.

4.3 Future Work and Goals

Having the two separate transport layers successfully merged, the only thing holding up the `aic94xx` driver from mainline acceptance is an unresolved bug. As of March 2006, a solution to this bug has been posted on the linux_scsi mailing list, so acceptance is very close.

Even after upstream acceptance, there will still be outstanding bugs. The very large list of problems that the team started with has dwindled down to a few, but issues remain. Plus, `aic94xx` is a new driver and we can expect to continue to uncover bugs as the driver gets used more heavily. It is the goal of the IBM LTC to make this driver stable and successful, so these efforts will continue.

5 Replacing a Proprietary Vendor Driver (`hostraid`)

Darrick Wong has been working with Adaptec to augment functionality and to add HostRAID support to `dmraid` as a replacement for a closed source module. This chapter examines the process by which he achieved that goal.

Adaptec HostRAID is an add-in BIOS component that attaches to various SCSI, SATA and Serial Attached SCSI controllers to provide bootable software RAID ("fakeraid") for entry-level RAID configurations. Like all fakeraids, the HostRAID component relies on a kernel driver to handle the actual I/O processing; this driver was only available in the form of the `a320raid` binary driver on Adaptec's Web site. The disadvantages of this approach are numerous: driver support exists only for a few distributions, bugs in binary modules are difficult to diagnose, version mismatches cause confusion, and the kernel becomes tainted. These issues cause enormous headaches for more than just customers; as Greg Kroah-Hartman pointed out at OSCON last year,[3] out-of-kernel drivers represent an ongoing maintenance problem for vendors as well, since a certified driver disk for a distribution often does not appear for anywhere between weeks and months after a distribution release. The response to all this, of course, is for somebody to write an open source driver and push it into the kernel.

[3] `http://www.kroah.com/linux/talks/oscon_2005_state_of_the_kernel/index.html`

5.1 Why `dmraid`?

It turns out that Adaptec attempted to write and submit an open-source driver a few years ago. The community discussed this "`emd`" driver but rejected it in favor of a different approach using device-mapper. The Adaptec developers disagreed with the proposed implementation, and ceased `emd` development. Due to continuing complaints about the lack of out-of-the-box HostRAID support in Linux, Mr. Wong decided in late 2005 to look into bridging the gap created by the abandonment of the `emd` driver; it seemed like it would not be difficult to transform the relevant parts of `emd` into a metadata format plugin for Heinz Mauelshagen's `dmraid` program. If successful, this represents a huge win for everybody—the support that was called for in the previous section is built without the need to write an entire RAID stack, and users get the support for which they have been clamoring! However, there were significant challenges even with this approach.

5.2 Convincing the Hardware Vendors

The first hurdle that had to be surmounted was selling `dmraid` as an `a320raid` replacement not only internally but also to Adaptec. As anticipated, the biggest challenges to the proposal was the suggestion that it would be easier to push Adaptec to maintain `a320raid`, and figuring out how to write this open source replacement without damaging the relationship between the two companies. Fortunately, the arguments presented by Mr. Kroah-Hartman[4] in favor of merging open source drivers upstream convinced management to go with the `dmraid` approach because everybody liked the prospect of Linux working out of the box in places where it previously did not. However, due to business realities, staffing limits and timing, it was difficult for Adaptec to commit the resources to complete the effort on its own. IBM, on the other hand, had a strong business case to reduce technical support load by augmenting `dmraid` and pushing it into the distributions. Thus it was decided that there was sufficient impetus at both companies to start a cooperative effort to get the `dmraid` work done.

However, that was only half the story—learning how the hardware works and building testing rigs can be a Herculean effort. In this case, though, it helped enormously that in addition to the `emd` source floating around in the Google® search engine and the IBM LTC's HostRAID-equipped test systems, Adaptec's engineers were available to answer questions about how the hardware actually operated. They were also indispensable in providing pointers to relevant metadata standards that went a long way towards revealing the intent behind how things worked. They also provided some sample hardware for testing purposes.

5.3 Writing and Pushing Code

Armed with specifications, Mr. Wong wrote the HostRAID format handler for `dmraid` and sent it around on the dm-devel and ataraid mailing lists for review. After a few rounds of improvements and testing, the basic handler code was incorporated into the 1.0.0rc10 release of `dmraid`. Acting as a mediating code-monkey between Adaptec and the `dmraid` developers has been a surprisingly straightforward process—Mr. Wong asks the hardware vendor what the software has to do to drive the hardware, asks the upstream maintainers how the code should integrate itself with the existing corpus, and produces something that (hopefully) satisfies both. Fortunately, `dmraid` is a userspace configuration program and not a core kernel component, which made debugging and

[4] See `Documentation/stable_api_nonsense.txt`.

acceptance easier. Small-scale stress and performance tests have not revealed any major regressions against the `a320raid` driver.

Despite the device-mapper code having been in the 2.6 kernel for quite a long time, there were still a few instances where code had to be submitted to the mainline kernel to make things work as smoothly as they did with the binary drivers. In the course of testing a full-system bringup with `dmraid`, it was discovered that device-mapper devices do not report disk geometry through the `HDIO_GETGEO` ioctl. This is not a problem for architectures that do not rely on disk geometries, but two notable geometry-dependent programs (fdisk and grub on i386 and x86_64 systems) can potentially trip over this omission. A patch was created fairly quickly and submitted to linux-kernel for discussion, with the first few iterations of the patches encountering objections for one reason or another. Eventually, the patches were reworked into an acceptable form, and they went into 2.6.17. Though this particular set of interactions did not involve Adaptec, it serves as a good example of how one can work with objectors even if one's first patch fails to gain traction; clearly, giving up would not have been the best option!

The second phase of development for HostRAID support was trickier to manage. In March 2006, Mr. Wong became more deeply involved in the development of the general `dmraid` framework to add support for hot array reconfiguration. Unlike the first phase, where the `dmraid` design was well established, and writing the software was an exercise of connecting the dots from the metadata specifications to the existing program, this portion made Mr. Wong negotiate directly with several engineers at Red Hat to get approval for the design of new features. As of April 2006, the hot reconfiguration work was still in progress.

5.4 Selling Distributions on the Solution

The last piece of the puzzle was perhaps the most difficult to set in place: getting major distributions to agree to integrate `dmraid` into their core package sets and the installer.

For a hardware vendor, this can be a daunting task. While it is true that there is only one Linux community, there are many distributions. Furthermore, the big distributions want to see third-party pieces like the `dmraid` patches integrated upstream before they will take the features, thus making the job of a hardware vendor difficult. Given the difficult patch examination discussed in section 3.2, Red Hat and Novell's insistence upon this point is not surprising. Furthermore, in terms of negotiating patches with distributions, the IBM LTC has a particularly advantageous channel to go to bat for the hardware vendors because people in the IBM LTC spend a lot of time talking to the distributions. However, this is a fine line to walk because those same Linux vendors do not necessarily want the increased support load. Because these low-end RAID solutions are often used as boot drives on systems and not as a bolt-on solution that can be configured after installation, there is an added burden that the distributions must be convinced that it is in their best interest to modify their installers to know how to use `dmraid`.

In practical terms, this meant talking to three big distributions: Red Hat for RHEL, Novell for SLES, and Ubuntu®. In the case of Red Hat, the process was easy—because Heinz Mauelshagen, the author of `dmraid`, works for Red Hat, the release managers for Fedora® Core were already familiar with the project. This familiarity decreased their resistance to accepting `dmraid` because Mr. Wong was supporting a project of Red Hat's and asking them to put it in their distributions, instead of creating a wad of code that they had never seen before, and asking them to incorporate it. As

of April 2006, `dmraid` support was being developed for a future Ubuntu release, talks were ongoing with Novell about SLES, and Red Hat was asked to incorporate the `dmraid` features of Fedora Core into their enterprise distribution.

5.5 Where Does the Project Go from here?

One critical question emerged from this driver writing effort—since Mr. Wong did a large chunk of the work, how could he avoid giving hardware vendors the impression that somebody at IBM will write their drivers for them, and instead convince them to take an active role in driver development? Obviously, there is a bit more political maneuvering and business case manipulation going on behind the scenes, but even on a purely technical level there are several reasons why it is still better for hardware vendors to write drivers themselves. First of all, Mr. Wong unfortunately did not have access to *all* of the documentation and design work that went into Adaptec's chips. Though it is his hope that there were no glaring deficiencies introduced into the HostRAID-related parts of the `dmraid` code, only Adaptec would really know the answer to that, and development would be far more efficient if there was no need for him to play middle-man. Hardware vendors have their own driver writing teams that *do* have easy access to the hardware designers; these teams ought to handle the majority of the technical work and pull IBM in as a catalyst to help them to negotiate with the code maintainers in the community and the distributions. Furthermore, the thrust of these driver writing efforts is to encourage the vendors to work with the community and the distributions and not to rely totally on others to write open drivers for them. Creating all-out replacements for binary drivers is a last resort when nothing else works.

Moreover, there are enough users of HostRAID asking for better Linux support that the existence of the HostRAID plugin for `dmraid` project should be a signal that Linux is not cutting into Adaptec's user base. More likely, good driver support would increase the number of HostRAID users. Nobody likes to have to feed driver disks to machines at install time; negative feedback about this tedium from Linux users will likely coax future server design teams to choose more Linux-friendly components so that everything just works, thus avoiding support problems before they happen. Finally, working with the community makes everyone happy; the code that was written is proof that hardware vendors really can engage people in the community, given the right catalyst.

On the technical side, there is plenty still to be done—the `a320raid` migration story is still a bit rocky, `dmraid` lacks full support for the Storage Networking Industry Alliance's Disk Data Format (DDF), which was an attempt to standardize metadata formats, and of course the code will benefit from more testing. It would also be useful to explore how the `dmraid` code can be used as a recovery mechanism—ideally, one could yank the drives from a machine with a failed HostRAID controller, put them into *any* machine running Linux, and still be able to recover data from the array. Lastly, a performance comparison between hardware RAID, `dmraid`, and the binary driver ought to be made.

6 Lessons Learned

In the process of helping independent hardware vendors (IHVs) to work with the Linux community and the distributions, and even developing code to support and better utilize our own hardware, several rules have become readily

apparent. These guidelines are not imposed by the community just to force changes or to obstruct participation; as Mr. Kroah-Hartman has emphasized in the past, these suggestions are made to help code submitters find more bugs, to help kernel hackers to find bugs, and to make drivers more maintainable if somebody else becomes responsible for the code. Here, then, are suggestions compiled by many people over the years:

- Please read the files in `Documentation/`.
- Do not roll unrelated fixes into one incohesive patch.
- There are others who use the same code and will be interested in the patches; do not assume that they will not help to improve the code.
- Someone may totally rewrite a patch submission. This is perfectly fine; what matters is the patch that gets committed, because that is what the customers use.
- Always test patches.
- Somebody will find esoteric code-breaking conditions that nobody anticipated.
- The world extends beyond i386 and x86_64 systems. Solve problems generically.
- The road to integration may be long and hard, but the code will be better because of that.
- IHVs need to maintain good relationships with distributions. This means that IHVs need to stay abreast of the distribution's release plans, ensure that any code that will be submitted for a release has already been approved upstream, and possibly even contribute sample hardware for testing.
- Code forks are good for developers with divergent goals if there are solid reasons for forking.
- Do not ask the community for input and ignore them. If a developer does not like what another has to say, that developer ought to talk to the objector. If a developer thinks that other person's proposal is impossible, that sentiment should be demonstrated with code.
- Wholesale rejections of code are not a reason to disappear. Just because people are not amenable to the proposed approach does not mean that they will be hostile to all approaches.
- Driver writers should inform others of what they are working on early to avoid wholesale rejections once the work is finished. The development process in place does not have one submission deadline; it instead works in an iterative fashion.
- Hardware manufacturers need to be involved with the development of their drivers.
- Having code committed upstream is a powerful endorsement when trying to convince distributions to take the code.
- Adapting business processes to work with the Linux community may be hard, but doing so opens the door to new customers who ignore products that lack Linux support and buy products that do.
- Open drivers extend the life of hardware.

- Legal departments can set up barriers to working with Linux, but even limited participation is better than none at all.

7 Unanswered Questions

As the clever reader may already have suspected, a task of this gargantuan scope raises perhaps as many questions as it resolves. On the technical side, it is not always clear who will maintain and enhance the code that has been written. As the `pcnet32` driver case illustrates, the work may be picked up by various individuals with itches to scratch; as the other three cases show, the driver maintenance job becomes a collaborative effort between several different groups. One scenario that has not been explored, however, is the case where somebody at IBM writes a rudimentary driver to get the process started and gradually pushes the development burden back to the hardware vendor. Depending on the vendor's ability to adapt to the community's development style, this is a transition that must be managed carefully—at a bare minimum, the legal issues of moving code ownership around will have to be sorted out, and the vendors will have to be trained to work in an environment where third parties have the power to reject their code. Furthermore, early engagement with the consumers of the code is a crucial ingredient to integration. As this paper has also illustrated, developing a driver without the agreement of the required maintainers usually results in patch rejection.

A second issue to arise from this arrangement is the question of what to do about customer support. When a support endpoint such as a distribution or a vendor owns the code, responsibility for that code naturally falls upon the endpoint; this breaks down when the authors are acting as an intermediary between distributions, vendors, and (potentially) other parties. As the intermediaries, IBM could, presumably, coordinate a concerted response, but the goal is to fix the driver support problems first, and to try to make the vendors responsible for the drivers after that.

8 Wrapup

Having blazed some of the trails towards the goal of helping developers, product managers, legal teams, *et al.* and hardware manufacturers to learn how to become participants in the Linux community, it is hoped that others who are trying to encourage third parties and hardware vendors to join the Linux movement can use our experiences as a guide for how to go about this process. Furthermore, it is hoped that the questions for which no satisfactory resolution has been found will catalyze debate as to what options will work. Jonathan Corbet's BOF session at this same OLS conference may shed some light on this. Perhaps future Open Driver fora will attract more than just developers to this question; for certain, driver writing efforts such as these will not always be the products of individual contributors going forward, which means that project leaders and lawyers at hardware manufacturers need to be educated about ways to co-operate with Linux.

We would also like to thank (in no particular order) the following people for their help with this paper and our other efforts: James Bottomley, Jeff Garzik, Greg Kroah-Hartman, and Andrew Morton; Rob Tarte and Tom Treadway of Adaptec; Craig Thomas of OSDL; Tom Coughlan, Alasdair Kergon, and Heinz Mauelshagen of Red Hat; and finally Mike Anderson, Pat Gaughen, Dave Hansen, Sheila Harnett, AJ Johnson, Chris McDermott, Paul McKenney, Ram Pai, and Russ Weight of IBM. To the many people who reviewed this paper, we also send our enormous thanks. Furthermore, it is very

likely that there are others who we have inadvertently failed to acknowledge; to all of you, we apologize for the omission and thank you for your efforts.

9 Legal Statements