

pdc-ultra promise 150tx2+ driver bug? hard lock/other crashiness

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2003-08/5086.html>

From: Fredrick Knieper (*derfk_at_sbcglobal.net*)

Date: 08/19/03

To: linux-kernel@vger.kernel.org

Date: Mon, 18 Aug 2003 23:12:46 -0500

(I'm not subscribed, please CC any replies and flames to me)

I'm bringing this up here because I noticed traffic in the past month on this driver being GPL'd and possibly finding its way into the kernel itself. I'm also hoping that someone here can think of something I've overlooked or something to try to test why this is happening, or has come across this and knows of a fix.

I have a dual athlon 2400 (real MPs) with 1GB of RAM in a MSI K7D Master motherboard. This worked great with no instability or freezes for some time, then one day the system overheated when the AC went out. When I restarted it, everything worked except the IDE controller built into the motherboard would not operate in any DMA modes, in Linux or Windows. I ordered a Promise SATA 150TX2plus to replace this, since it has a PATA interface built in. After some work I managed to build the 1.00.0.8 drivers provided by Promise, and Linux would boot from the SATA controller. This worked for about two days before the system locked up hard. I thought "Well, maybe its the parallel IDE interface" so I purchased an PATA->SATA converter, plugged it in, and booted. Again, it ran for a while then froze hard. I thought "well, maybe its the converter?" so I purchased a 250GB SATA drive. (side note: I can access the full drive in Linux, and can confirm the data is correctly placed in windows 2000. Did I misread the messages saying that I should be unable to access past 137GB?). Now, something interesting happened. I discovered that I was able to repeatedly freeze the system by copying /usr from one drive to another. Intrigued I found out that I can produce the following behavior repeatedly, in various kernels from 2.4.20 to 2.4.22-rc1 with both the 1.00.0.8 driver on Promise's site, and the 1.00.0.10 driver linked here last month:

Using vmstat 1, I can watch as cache consumes nearly all available RAM. When about 6-10MB of free RAM is available, memory consumption slows and then holds steady for a few seconds, then continues until about 4-8MB of RAM remains, at which time one of the following three things happens (side note: the system **never** used the swap partition every time I tried this. The swap partition is also residing on the SATA controller. I tried using a swapfile,

Linux–Kernel: pdc–ultra promise 150tx2+ driver bug? hard lock/other crashiness

loop0 attached to a swapfile, and finally no swap and received the same results in each case)

1 – If I am using X, I almost always (90%) obtain a hard lock with capslock and scroll lock lights lit on my keyboard. The remaining 10% of the time, either #2 or #3 occurs.

2 – If I am writing to the drive, the system freezes hard. No messages at all are printed to the console, and the caps/scroll lock lights stay off. The filesystems (all either ext3fs or ext2fs) are more–or–less OK. I can repeat this with "dd if=/dev/zero of=/mnt/scratch/foo" and letting it run until the system freezes. The resulting filesize of foo seems to depend on how much memory is in use outside of cache and how much was flushed to disk, as it varies each time I repeat this, but is roughly between 970MB and 1100MB.

3 – If I am reading from the drive, the system goes into what I call "slow death" mode. The e100 driver is the first to die, all networking just stops (confounding attempts to log whats going on remotely). Shortly afterwards, the OCHI driver quits running the mouse (ps/2 keyboard still functions). After this, the kernel starts dumping ext3fs errors for bad directory inodes and errors for attempts to access beyond the end of the device (this is with both a 80GB drive and the 250GB drive, mind you). At this point, the entire filesystem across *all* rw mounted partitions becomes corrupt. I can no longer run most executables (oddly, I can still run "ls" and "cat", yet they are not bash builtins) and many files are replaced by binary garbage (I tried to look into the logs to see if any useful information was present. What's worse, if I do not immediately reboot, it appears that the kernel actually begins to flush the corrupt cache back to the drive... the last time I spent maybe 15 minutes looking to see if I could get useful log information out of the system, when I did finally hard reboot the computer (the shutdown program could not be run), fsck.ext3 detected no problems on a journal check with /usr, but my /usr/include and /usr/share directories were both corrupted (a full fsck.ext2 reattached the subdirectories into lost+found). This did not happen any other time I experienced this behavior, but every other time I rebooted the system immediately. To replicate this, I can execute "find /usr -exec cat \{\} \; > /dev/null".

In working out what was going on, I tried several things. First, I believed I had found a bug in ext3fs that had somehow escaped detection despite massive usage. So I created a scratch ext2fs partition and found I could repeat the symptoms on it. I could NOT repeat the crash over NFS (in fact, when using NFS, the swap partition on the SATA controller was actually used). I tried compiling the kernel with and without SMP, with highmem IO on and off, and with 4GB highmem support and no highmem support (also recompiled the sata driver each time). Under each of these configurations the crash would occur. The crash does NOT occur when writing to the drive directly, via "dd if=/dev/zero of=/dev/sdb1" The computer will happily fill the entire partition without freezing. I found that I can buy more time by unmounting the affected partitions (which returns the filesystem cache to free RAM).

Linux-Kernel: pdc-ultra promise 150tx2+ driver bug? hard lock/other crashiness

After checking the lkml archives for similar problems, I came across a post about the NMI watchdog. I tried booting the kernel with the `nmi_watchdog=1` parameter, and received a boot message reading:

```
testing NMI watchdog ... CPU#0: NMI appears to be stuck!
```

after which everything proceeded normally. This had no apparent effect and produced no further output when the system froze. The documentation seems to indicate that the oops part requires a (seperate?) boot parameter, but `nmi_watchdog.txt` neglects to mention what it is (unless the `nmi_watchdog=n` is the enabling parameter).

I've also tried running `memtest86` on all tests for 10 passes, after which I'm fairly certain its not the memory or `cpu0`'s cache or processor. Is 10 enough for me to be certain about this or should I let it run longer before I make this claim?

So now, any suggestions or hints on debugging this? Given that it only happens when the filesystem cache for a SATA partition fills up, it seems that its something that happens when the kernel decides that its time to write dirty cache to disk or to abandon some old cache in favor of new data. This is, of course, my uninformed opinion on the subject, given that I do not understand how Linux handles its filesystem cache, or how the SATA driver works.

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>