

Re: o_sync in vfat driver

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-02/msg09981.html>

- *From:* "linux-os (Dick Johnson)" <linux-os@xxxxxxxxxxxxx>
 - *Date:* Tue, 28 Feb 2006 11:16:30 -0500
-

On Tue, 28 Feb 2006, Lennart Sorensen wrote:

On Tue, Feb 28, 2006 at 08:10:44AM -0500, linux-os (Dick Johnson) wrote:

On Mon, 27 Feb 2006 col-pepper@xxxxxxxxxxxxx wrote:

On Mon, 27 Feb 2006 22:32:07 +0100, linux-os (Dick Johnson)
<linux-os@xxxxxxxxxxxxx> wrote:

Flash does not get zeroed to be written! It gets erased, which sets all the bits to '1', i.e., all bytes to 0xff.

Thanks for the correction, but that does not change the discussion.

Further, the designers of flash disks are not stupid as you assume. The direct access occurs to static RAM (read/write stuff).

I'm not assuming anything . Some hardware has been killed by this issue.

<http://lkml.org/lkml/2005/5/13/144>

No. That hardware was not killed by that issue. The writer, or another who had encountered the same issue, eventually repartitioned and reformatted the device. The partition table had gotten corrupted by some experiments and the writer assumed that the device was broken.

Re: o_sync in vfat driver

It wasn't.

Also, if you read other elements in this thread, you would have learned about something that has become somewhat of a red herring.

It takes about a second to erase a 64k physical sector. This is a required operation before it is written. Since the projected life of these new devices is about 5 to 10 million such cycles, (older NAND flash used in modems was only 100–200k) the writer would have to be running that "brand new device" for at least 5 million seconds. Let's see:

How come I can write to my compact flash at about 2M/s if you claim it takes 1s to erase a 64k sector? Somehow I think your number is much too high. Or it can do multiple erases at the same time.

Also the 5 to 10 million is a lot higher than the numbers the makers of the compact flash cards I use claim.

Here is an instrumented erase function on a driver that rewrites the first sector of a BIOS ROM. Unlike the Flash DISKS, the BIOS ROM has no buffering in static RAM so you can gustomate the actual time to erase.....

```
//-----  
//  
// This erases a page and waits for the erasure to complete. It  
// returns false if it failed.  
//  
static int erase(void *bios, int page)  
{  
    int era;  
    flags_t flags;  
    jiffie_t ticks, start;  
    spin_lock_irqsave(&info->lock, flags);  
    erase_page(bios, page);  
    spin_unlock_irqrestore(&info->lock, flags);  
  
    start = jiffies;  
  
    ticks = jiffies + (ERA_TIME * HZ);  
    era = 0x00;  
    while(time_before(jiffies, ticks))  
    {  
        if((era = check_erase(bios, page)))  
            break;  
        if(signal_pending(current))  
            break;  
    }  
}
```

Re: o_sync in vfat driver

```
set_current_state(TASK_INTERRUPTIBLE);
schedule_timeout(1);
}
set_current_state(TASK_RUNNING);
```

```
printk("They don't believe... %d\n", (int) (jiffies - start));
```

```
return era;
}
```

[SNIPPED...]

On the system I rewrite a BIOS sector on, jiffies is 1024 ticks/second.

```
parport: PnPBIOS parport detected.
parport0: PC-style at 0x378, irq 7 [PCSP,TRISTATE]
lp0: using parport0 (interrupt-driven).
lp0: console ready
device eth0 entered promiscuous mode
device eth0 left promiscuous mode
device eth0 entered promiscuous mode
device eth0 left promiscuous mode
Analogic-BiosDev : Initialization complete
They don't believe... 1004
```

Now, the wait for erase always sleeps for at least a timer-tick (about a milisecond) so this may take longer than the physical erase, but not much longer.

The erase function is:

```
#-----
#
# This erases the NVRAM page (block). It doesn't wait for completion.
# Each block is 64k in length.
# M29W040B chip
#
.section .text
erase_page:
pushl %ebx
movl BUF(%esp), %ebx # Address of the chip
movl DAT(%esp), %ecx # The page
andl $0x07, %ecx # Max pages
shll $0x10, %ecx # Times 64k
movb $0xf0, (%ebx) # Reset
movb $0xaa, 0x555(%ebx)
movb $0x55, 0x2aa(%ebx)
movb $0x80, 0x555(%ebx)
movb $0xaa, 0x555(%ebx)
movb $0x55, 0x2aa(%ebx)
```

Re: o_sync in vfat driver

Re: o_sync in vfat driver

```
movb $0x30, (%ecx,%ebx)
popl %ebx
ret
.size erase_page,.-erase_page
.type erase_page,@function
.global erase_page
```

And the check-erase function is this:

```
#-----
#
# This reads the whole M29W040B page, looking for all 0xffff words.
# It returns non-zero if it has been erased and zero otherwise.
#
check_erase:
pushl %edi
movl BUF(%esp), %edi # Point to buffer
movl DAT(%esp), %eax # 64k page
andl $0x07, %eax # Max pages possible
shll $0x10, %eax # Times 64k
addl %eax, %edi # Offset to start
cld
movl $0x8000, %ecx # Number of words to check
movl $-1, %eax # What to look for
repz scasw # Look for all 0xffff
jz 1f # All erased
incl %eax # -1 becomes zero
1: popl %edi
ret
.size check_erase,.-check_erase
.type check_erase,@function
.global check_erase
```

60 seconds per minute
3600 seconds per hour
86400 seconds per day.

$5,000,000 / 86400 = 57$ days of continuous writes to the same sector. The writer surely would have a strange file because he states that even a single large file can destroy the drive if it is mounted with the "sync" option.

Also, the failure mode of NAND flash is not that it becomes "destroyed". The failure mode is a slow loss of data. The devices no longer retain data for a zillion years, only a few hundred, eventually, only a year or so. So when somebody claims that the flash has gotten destroyed, they need to have

Re: o_sync in vfat driver

written it for a few months, then waited for a few years before reporting the event.

Some flash devices can be "destroyed" by loosing power in the middle of a write, since this causes them to corrupt their table of blocks and only the manufacturer has the ability to reset that. Fortunately this doesn't seem like too common a design.

```
# dd if=/dev/zero of=/dev/whatever bs=1M count=128
```

Fixes a 128 megabyte flash disk, plug in other values for other sizes.

Len Sorensen

Cheers,

Dick Johnson

Penguin : Linux version 2.6.15.4 on an i686 machine (5589.54 BogoMips).

Warning : 98.36% of all statistics are fiction, book release in April.

—

The information transmitted in this message is confidential and may be privileged. Any review, retransmission, dissemination, or other use of this information by persons or entities other than the intended recipient is prohibited. If you are not the intended recipient, please notify Analogic Corporation immediately – by replying to this message or by sending an email to DeliveryErrors@xxxxxxxxxxxxx – and destroy all copies of this information, including any attachments, without reading or disclosing them.

Thank you.

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>