Adventures in Database Corruption

Danny Robson

Warning

Some of the following commands have the potentially to totally destroy your filesystem.

I almost did this twice.

Be careful.

Caveats

This occurred in May. Some details were reconstructed from memory.

Context

I write analysis tools for a social media site recreationally.

Context

- 87M comments
- ~350GB database
- ~600b / comment

Oh no...

```
reddit=> vacuum comment;
ERROR:  could not read block 13426454 in file
    "base/16854/90676088.102": Input/output error
```

Oh no.....

```
kernel: bcache: bch_count_backing_io_errors() md01:
    IO error on backing device, unrecoverable
```

Oh no.......

3 month old backup...

First step

Lots of swearing.

Second step

```
systemctl stop postgresql-14.service
```

Backup as much as possible. **Right now**.

Potential loss

- 3 months history
- 4.5M comments
- Probably external casualties.

Probing the disks

Which **exact** disks have failed?

- `smartctl`
- `mdstat`
- `dd`

smartctl

Check SMART status of each disk.

```
sybil ~ # smartctl -H /dev/sda
smartctl 7.3 2022-02-28 r5338 [x86_64-linux-6.1.6-gentoo] (local
Copyright (C) 2002-22, Bruce Allen, Christian Franke, www.smartmo

=== START OF READ SMART DATA SECTION ===
SMART overall-health self-assessment test result: PASSED
```

mdstat

Scrub entire RAID5 array.

```
checkarray --all
sleep 1d
cat /proc/mdstat
```

Worth tailing syslog.

`mdstat`

No issues.

I don’t undestand, but I’m intrigued…

Poking files

Lets try to read the damaged components directly.

Brute force

```
sybil ~/ # cp 90676088.102.orig /dev/null
cp: error reading '90676088.102.orig': Input/output error
```

```
sybil kernel: bcache: bch_count_backing_io_errors()
    md69: Read-ahead I/O failed on backing device, ignore
sybil kernel: bcache: bch_count_backing_io_errors()
    md69: IO error on backing device, unrecoverable
```

Finesse

```
ERROR:  could not read block 13426454 in file
    "base/16854/90676088.102": Input/output error
```

Finesse

Calculate bad offset in the file.

```
block = 13426454

pg_sector = 8192
pg_chunk = 1GB = 1024*1024*1024

offset = (block * pg_sector) % pg_chunk
       = 467845120
```

Finesse

```
sybil ~/recover/orig # dd if=90676088.102 \
    bs=1 skip=467845120 count=8192 \
    of=/dev/null
dd: error reading '90676088.102.orig': Input/output error
0+0 records in
0+0 records out
0 bytes copied, 0.0258018 s, 0.0 kB/s
```

Poking disks

Let’s try to read from the disks directly

bcache

Prevent the disk from dropping out on us

```
echo 999999 > /sys/block/bcache0/bcache/io_error_limit
```

xfs_bmap

```
extent: [startoffset..endoffset]: startblock..endblock
```

```
sybil ~/ # xfs_bmap 90676088.102
90676088.102:
        0: [0..2097151]: 323562496..325659647
```

- 512b block size units

Maths

```
file_offset = 467845120/512
            = 913760

file_base = 323562496

lvm_offset = $(pvs --noheadings \
    -o pe_start --units 512b \
    /dev/bcache0) = 2048

bcache_offset = 8192/512 = 16

target = $bcache_offset + $lvm_offset + \
    $file_base + $file_offset = 324478304
```

dd - **Success?**

```
sybil ~/ # dd if=/dev/bcache0 bs=512 \
    skip=$((913760+2048+323562496)) \
    count=16 of=/dev/zero
16+0 records in
16+0 records out
8192 bytes (8.2 kB, 8.0 KiB) copied, 0.000600769 s, 13.6 MB/s
```

dd - **Failure**

```
sybil kernel: bcache: bch_count_backing_io_errors() md69:
    IO error on backing device, unrecoverable
sybil kernel: Buffer I/O error on dev bcache0,
    logical block 40559790, async page read
```

Further

It should be possible to delve further to uncover which disk was corrupted.

However, I was impatient, so... onwards...

"Recovery"

What if we’re happy to lose the damaged records?

ddrescue

```
ddrescue 90676088.102 90676088.102.new mapfile
# DO NOT DO THIS
ddrescue --fill-mode=- <(printf "BADSECTOR") \
    90676088.102.new mapfile
```

ddrescue: mapfile

```
# Mapfile. Created by GNU ddrescue version 1.27
# Command line: ddrescue --fill-mode=- /dev/fd/63 90676088.102 ma
# Start time:   2023-05-18 13:05:14
# Current time: 2023-05-18 13:05:14
# Finished
# current_pos  current_status  current_pass
0x1C0C4000     +               1
#      pos        size  status
0x00000000  0x1BE00000  +
0x1BE00000  0x00080000  -
0x1BE80000  0x00244000  +
0x1C0C4000  0x00004000  -
0x1C0C8000  0x23F38000  +
```

Damage

- 528kb unreadable.
- Under 1000 records.

postgresql

```
SET zero_damaged_pages = on;
vacuum full comment;
reindex table comment;
SET zero_damaged_pages = off;

# select blocks_done::float/blocks_total
#   from pg_stat_progress_create_index;
```

Lucky other tables didn’t depend on it...

Lessons

- Don’t neglect backups
- Don’t neglect scrubs
- Don’t neglect whole system tests

Future

- Attribute logical sectors to disks.
- Figure out why `mdadm` didn’t see any errors.

Thanks