When Solid State Drives are not that solid

Adam Surak | Jun 15th 2015 | 12 min read | Engineering

It looked just like another page in the middle of the night. One of the servers of our search API stopped processing the indexing jobs for an unknown reason. Since we build systems in Algolia for high availability and resiliency, nothing bad was happening. The new API calls were correctly redirected to the rest of the healthy machines in the cluster and the only impact on the service was one woken-up engineer. It was time to find out what was going on.

SUMMARY:

1) the issue raised by Algolia is due to a Linux kernel error

2) Linux kernel error can affect any SSD under the same operating conditions

3) Samsung has also posted a Linux kernel patch that should fix the issue

UPDATE June 16:

A lot of discussions started pointing out that the issue is related to the newly introduced queued TRIM. This is not correct. The TRIM on our drives is un-queued and the issue we have found is not related to the latest changes in the Linux Kernel to disable this feature.

> # smartctl -l gplog,0x13 /dev/sda
> smartctl 6.2 2013-07-26 r3841 [x86_64-linux-3.16.0-31-generic] (local build)
> Copyright (C) 2002-13, Bruce Allen, Christian Franke, www.smartmontools.org
>
> General Purpose Log 0x13 does not exist (override with '-T permissive' option)

UPDATE June 17:

We got contacted by Samsung and we provided them all the system specifications and all the information about the issue we had. We will continue to provide Samsung all the necessary information in order to resolve the issue.

UPDATE June 18:

We just had a conference call with the European branch and the Korean HQ of Samsung. Their engineers are going to visit one of the datacenters we have servers in and in cooperation with our server provider they will inspect the mentioned SSDs in our SW and HW setup.

UPDATE June 19:

On Monday June 22, the engineering team from Samsung is going analyze

one of our servers in Singapore and if nothing is found on-site, the server will travel to Samsung HQ in Korea for further analysis.

UPDATE July 13:
Since the last update of this blog-post, we have been in a cooperation with Samsung trying to help them find the issue, during this investigation we agreed with Samsung not to communicate until their approval.

As the issue was not reproduced on our server in Singapore, the reproduction is now running under Samsung supervision in Korea, out of our environment. Although Samsung requested multiple times an access to our software and corrupted data, we could not provide it to them in order to protect the privacy and data of our customers.

Samsung asked us to inform you about this:

- Samsung tried to duplicate the failure with the latest script provided to them, but no single failure has been reproduced so far.

- Samsung will do further tests, most likely from week 29 onwards, with a much more intensive script provided by Algolia.

After unsuccessful tries to reproduce the issue with Bash scripts we have decided to help them by creating a small C++ program that simulates the writing style and pattern of our application (no files are open with O_DIRECT). We believe that if the issue is coming from a specific way we are using the standard kernel calls, it might take a couple of days and terabytes of data to be written to the drive. We have been informed by Samsung that no issue of this kind have been reported to them. Our server provider has modified their Ubuntu 14.04 images to disable the fstrim cron in order to

avoid this issue. For the last couple of months after not using trim anymore we have not seen the issue again.

UPDATE July 17:
We have just finished a conference call with Samsung considering the failure analysis of this issue. Samsung engineering team has been able to successfully reproduce the issue with our latest provided binary.

Samsung had a concrete conclusion that the issue is not related to Samsung SSD or Algolia software but is related to the Linux kernel.

Samsung has developed a kernel patch to resolve this issue and the official statement with details will be released tomorrow, July 18 on Linux community with the Linux patch guide. Our testing code is available on GitHub.

This has been an amazing ride, thank you everyone for joining, we have arrived at the destination.

For all followers of this blogpost and all the new readers:

The discovered issue has much bigger impact than we originally expected and is not caused by Samsung SSDs, as we originally assumed.
My personal apologies to Samsung!

The NGINX daemon serving all the HTTP(S) communication of our API was up and ready to serve the search queries but the indexing process crashed.

Since the indexing process is guarded by supervise, crashing in a loop would have been understandable but a complete crash was not. As it turned out the filesystem was in a read-only mode. All right, let's assume it was a cosmic ray 🙂 the filesystem got fixed, files were restored from another healthy server and everything looked fine again.

The next day another server ended with filesystem in read-only, two hours after another one and then next hour another one. Something was going on. After restoring the filesystem and the files, it was time for serious analysis since this was not a one time thing. At this point, we did a breakdown of the software involved in our storage stack and went through the recent changes.

Investigation & debugging time!

We first asked ourselves if it could be related to our software. Are we using non-safe system calls or processing the data in an unsafe way? Did we incorrectly read and write the files in the memory before flushing it to disk?

- Filesystem – Is there a bug in ext4? Can we access the memory space of allocation tables by accident?
- Mdraid – Is there a bug in mdadm? Did we use an improper configuration?
- Driver – Does the driver have a bug?
- SSD – Is the SSD dying? Or even worse, is there a problem with the firmware of the drive?

We even started to bet where the problem was and exactly proposed, in this order, the possible solutions going from easy to super-hard.

Going through storage procedures of our software stack allowed us to set up traps and in case the problem happens again, we would be able to better isolate the corrupted parts. Looking at every single storage call of our engine gave us enough confidence that the problem was not coming from the way in which we manipulate the data. Unfortunately.

One hour later, another server was corrupted. This time we took it out of the cluster and started to inspect it bit by bit. Before we fixed the filesystem, we noticed that some pieces of our files were missing (zeroed) – file modification date was unchanged, size was unchanged, just some parts were filled with zeros. Small files were completely erased. This was weird, so we started to think if it was possible that our application could access certain portions of the memory where the OS/filesystem had something mapped because otherwise our application cannot modify a file without the filesystem noticing. Having our software written in C++ brought a lot of crazy ideas of what happened. This turned out to be a dead-end as all of these memory blocks were out of our reach.

So is there an issue in the ext4? Going through the kernel changelog looking for ext4 related issues was a terrifying experience. In almost every version we found a fixed bug that could theoretically impact us. I have to admit, I slept better before reading the changelog.

We had kernels 3.2, 3.10, 3.13 and 3.16 distributed between the most often corrupted machines and waited to see which of the mines blows up. All of them did. Another dead-end. Maybe there was an issue in ext4 that no one else has seen before? The chance that we were this “lucky” was quite low and we did not want to end up in a situation like that. The possibility of a bug in ext4 was still open but highly improbable.

What if there was an issue in mdadm? Looking at the changelog gave us confidence that we should not go down this path.

The level of despair was reaching a critical level and the pages in the middle of the night were unstoppable. We spent a big portion of two weeks just isolating machines as quickly as possible and restoring them as quickly as possible. The one thing we did was to implement a check in our software that looked for empty blocks in the index files, even when they were not used, and alerted us in advance.

Not a single day without corruptions

While more and more machines were dying, we had managed to automate the restore procedure to a level we were comfortable with. At every failure, we tried to look at different patterns of the corruption in hopes that we would find the smallest common denominator. They all had the same characteristics. But one thing started to be more and more clear – we saw the issue only on a portion of our servers. The software stack was identical but the hardware was slightly different. Mainly the SSDs were different but they were all from the same manufacturer. This was very alarming and led us to contact our server provider to ask if they have ever seen something like this before. It's hard to convince a technical support person about a problem that you see only once in a while, with the latest firmware and that you cannot reproduce on demand. We were not very successful but at least we had one small victory on our side.

Knowing that the issue existed somewhere in the combination of the software and drive itself, we reproduced the identical software stack from our servers with different drives. And? Nothing, the corruption appeared again. So it was quite safe to assume the problem was not in the software stack and was more drive related. But what causes a block to change the

content without the rest of the system noticing? That would be a lot of rotten bits in a sequence...

The days started to become a routine – long shower, breakfast, restoring corrupted servers, lunch, restoring corrupted servers, dinner, restoring corrupted servers. Until one long morning shower full of thinking, "how big was the sequence?" As it turned out, the lost data was always 512 bytes, which is one block on the drive. One step further, a block ends up to be full of zeroes. A hardware bug? Or is the block zeroed? What can zero the block? TRIM! Trim instructs the SSD drive to zero the empty blocks. But these block were not empty and other types of SSDs were not impacted. We gave it a try and disabled TRIM across all of our servers. It would explain everything!

The next day not a single server was corrupted, two days silence, then a week. The nightmare was over! At least we thought so... a month after we isolated the problem, a server restarted and came up with corrupted data but only from the small files – including certificates. Even improper shutdown cannot cause this.

Poking around in the source code of the kernel looking for the trim related code, we came to the trim blacklist. This blacklist configures a specific behavior for certain SSD drives and identifies the drives based on the regexp of the model name. Our working SSDs were explicitly allowed full operation of the TRIM but some of the SSDs of our affected manufacturer were limited. Our affected drives did not match any pattern so they were implicitly allowed full operation.

The complete picture

At this moment we finally got a complete picture of what was going on. The system was issuing a TRIM to erase empty blocks, the command got

misinterpreted by the drive and the controller erased blocks it was not supposed to. Therefore our files ended-up with 512 bytes of zeroes, files smaller than 512 bytes were completely zeroed. When we were lucky enough, the misbehaving TRIM hit the super-block of the filesystem and caused a corruption. After disabling the TRIM, the live big files were no longer corrupted but the small files that were once mapped to the memory and never changed since then had two states – correct content in the memory and corrupted one on the drive. Running a check on the files found nothing because they were never fetched again from the drive and just silently read from the memory. Massive reboot of servers came into play to restore the data consistency but after many weeks of hunting a ghost we came to the end.

As a result, we informed our server provider about the affected SSDs and they informed the manufacturer. Our new deployments were switched to different SSD drives and we don't recommend anyone to use any SSD that is anyhow mentioned in a bad way by the Linux kernel. Also be careful, even when you don't enable the TRIM explicitly, at least since Ubuntu 14.04 the explicit FSTRIM runs in a cron once per week on all partitions – the freeze of your storage for a couple of seconds will be your smallest problem.

TL;DR

~~Broken SSDs~~: (Drives on which we have detected the issue)

- SAMSUNG MZ7WD480HCGM-00003
- SAMSUNG MZ7GE480HMHP-00003
- SAMSUNG MZ7GE240HMGR-00003
- Samsung SSD 840 PRO Series
  recently blacklisted for 8-series blacklist

- Samsung SSD 850 PRO 512GB
  recently blacklisted as 850 Pro and later in 8-series blacklist

~~Working SSDs~~: (Drives on which we have NOT detected the issue)

- Intel S3500
- Intel S3700
- Intel S3710

ABOUT THE AUTHOR

Adam Surak

Brian Bulkowski • 3 years ago

At Aerospike, we have a NoSQL database that's been run on a lot of flash drives, for years, and we've found a few things:
* Don't use TRIM. There are too many bad controllers that (at best) do bizarre performance actions. TRIM should be a good thing.... but it's (apparently) too complex for most controller writers.
* Don't use a file system. Aerospike has its own native data layout, and you can use files, but lousy things happen.
* Some manufacturers go through bad times, so keep testing. We built a tool http://github.com/aerospike... so we can prove to manufactures when they have bugs. If I tried to list all the devices that we tested and had firmware issues (either performance or "functionality") it would be a long and embarrassing list. We have over 4 years of data covering a lot of manufacturers.

24 • Reply • Share ›

Adam Surak ➚ Brian Bulkowski • 3 years ago

I totally agree, you should not use TRIM on enterprise SSDs. The idea to run without filesystem is interesting as we are not that far from it. Thank you for the testing tool, that's very impressive! I'll give it a try on our HW. Do you share the results somewhere?

2 • Reply • Share ›

Michael Böckling ➚ Adam Surak • 3 years ago

Its a bit hidden, but they do:
http://www.aerospike.com/do...

8 • Reply • Share ›

CustomDesigned ➚ Michael Böckling • 3 years ago

That list only shows bad performance - which is not a show stopper for a home user. We are frightened by the prospect of TRIM trimming the wrong blocks, however.

3 • Reply • Share ›

Roman Ovchinnikov ➚ Brian Bulkowski • a year ago

can you describe a bit more, why you strictly against using FS?
> Don't use a file system. Aerospike has its own native data layout, and you can use files, but lousy things happen.

We are in the process of evaluating FS improvements for reads - https://discuss.aerospike.c... , for now looks promising

^ v • Reply • Share ›

Theodore Ts'o • 3 years ago

Nearly all of the ext4 bugs that have been fixed in the last couple of releases are ones which were around for years, and very hard to trigger (in one case we fixed a bug in ext4 that was also in ext3, and had survived multiple rounds of enterprise linux release testing by IBM, HP, Red Hat, and SuSE), or are ones which were introduced during the merge window and for which the fix was quickly caught and pushed out before the kernel was released. The sort of bugs that we worry about are ones where you have blocks that are being written using async I/O and direct I/O racing with the same blocks getting allocated or truncated or punched out using the new punch hole functionality. Most applications don't have these sorts of extreme workload characteristics; they just append to a file, or rewrite blocks that were already allocated.

In particular, if you are using a standard 4k block file system, with the default mkfs options, I really wouldn't worry too much. The bugs tend to happen in the newer, more esoteric file system features, such as inline data, bigalloc file systems (especially if you are doing random sparse writes tracing with truncates/punch hole operaitons), etc. But that's not how most people use ext4.

If it will make you feel any better about file system bugs, we run very careful regression testing during the entire file system development process, and if a commit causes a regression, I'll drop the patch. And the regression tests are generally far more rigorous than typical file system workloads.

19 ^ v • Reply • Share ›

Adam Surak ➚ Theodore Ts'o • 3 years ago

I still have great confidence in the work you do and great respect for it

2 ^ v • Reply • Share ›

Heinz Kurtz ➚ Adam Surak • 3 years ago

Any news?

^ v • Reply • Share ›

Karol "Zal" Zalewski • 3 years ago

Consumer grade Samsung SSDs with newest firmware declare that they support queued TRIM. The funny thing is that they don't do that [1] - they lockup (power down is needed). And now you say that other Samsung disk do similar thing for un-queued TRIM? Scary thing...

[1] https://bugs.launchpad.net/...

7 ^ v • Reply • Share ›

issafram • 3 years ago

What is the latest update after the Samsung visit?

6 ^ v • Reply • Share ›

This comment was deleted.

Heinz Kurtz ➚ Guest • 3 years ago

The Linux world should definitely tone down on the TRIM. There's too much of it everywhere.

mkfs shouldn't trim. If I wanted it to do that, I could always use blkdiscard first.

1 ^ v • Reply • Share ›

This comment was deleted.

Heinz Kurtz ➚ Guest • 3 years ago

Have you ever used photorec? Never? Honestly? Good for you. Good for everyone else? People make mistakes. On HDD you can revert some of them. On SSD it's game over, cause SSD happily throw away gigabytes of data in an eyeblink.

It's may be worth the extra performance and lifetime on a hot database server. For the average Linux User it may be different.

I'm not asking to remove functionality. I just wish for saner defaults.

fstrim offers best of both worlds. TRIM yes, no long term performance degradation, but it's not instant so there's still time to disable the weekly/monthly fstrim cron job after sending the wrong rm command.

But that only works if other programs stop issuing TRIM without asking you.

2 ^ | v • Reply • Share ›

Hugo Vinícius ➔ Heinz Kurtz • 3 years ago

I've heard of problems involving the TRIM command, but I've never gave any importance to them. Now, reading this article, I'll be more cautious about it, even because I have two 850 Pros and one 850 EVO.

For recovery of accidental deleted files, I always turn to File History in my Windows 8 machine. Also, all my data are on SSDs, but with periodic, manual copies to HDs and my laptop. Since then, I don't remember of losing a file anymore.

PS: sorry for my English. It isn't my native language.

3 ^ | v • Reply • Share ›

Dooshe Nozzle ➔ Hugo Vinícius • 3 years ago

Your English is quite good. No need to apologize.

1 ^ | v • Reply • Share ›

Diogo Carvalho • 3 years ago

Complete non-savvy user here, but very concerned end-user on Mac OS X with trim force enable via Trim enabler, on a Fusion drive with one Samsung 840 PRO 256Gb. Am I on the verge of finding my life work corrupted? I've disable trim now, but have extensive backups that should have permeated the possibly corrupted zeroed files on them. Please advise.

4 ^ | v • Reply • Share ›

Adam Surak ➔ Diogo Carvalho • 3 years ago

Hard to say how Mac OS X behaves, and mainly in Fusion drive. Definitely keep backups and if you come to a weird file, take a look for zeroed blocks.

4 ^ | v • Reply • Share ›

Diogo Carvalho ➔ Adam Surak • 3 years ago

Thanks for the input, but won't the possibly zeroed affected files be permeated to the backups? How do I

check for zeroed blocks?

2 ^ | v • Reply • Share ›

Theo Brinkman ➚ Diogo Carvalho • 3 years ago

If you use TimeMachine for backups,you'll be able to roll back to versions *before* the zeroed block happened. To *see* the zeroed block, you'll need to open the file in a hex editor, or something similar. A block of 512 bytes worth of '00' will stand out pretty easily.

3 ^ | v • Reply • Share ›

Heinz Kurtz ➚ Diogo Carvalho • 3 years ago

> Am I on the verge of finding my life work corrupted?

Anyone who does not have backups is on that verge. And Apple's idea of not enabling trim is understandable when it's a) often buggy and b) not really as important as people make it out to be. SSD work well without TRIM.

2 ^ | v • Reply • Share ›

You are wrong ➚ Heinz Kurtz • 3 years ago

Backups don't save you from file corruption.

2 ^ | v • Reply • Share ›

Heinz Kurtz ➚ You are wrong • 3 years ago

But they do, unless you're cracknob enough to delete your old backups in favour of new ones, thereby replacing intact backups with corrupt ones...

1 ^ | v • Reply • Share ›

This comment was deleted.

Heinz Kurtz ➚ Guest • 3 years ago

As far back as possible. I prefer incremental backups that just add new/changed files instead of re-backing up the same files over and over again. It's not going to use more or less storage if you overwrite your old intact copy with a new corrupted one.

Maybe I'm too oldfashioned though. I still don't use ZFS/btrfs. ;)

For small static data (photos) I make extra

copies to optical media. At least they survive when lightning hits while your entire HDD zoo is working on those backups...

1 ^ | v • Reply • Share ›

Rob Cadwallader ➚ Heinz Kurtz • 3 years ago

I think this is an important distinction that a lot of people don't understand. TRIM is a hook that allows an OS to invoke garbage collection (usually through the recycle bin/trash can function in the OS). Disabling TRIM in the OS doesn't disable garbage collection, it simply diverts the garbage collection to the firmware on the SSD controller instead of being invoked at the OS level.

1 ^ | v • Reply • Share ›

SSD-User • 3 years ago

So, it's July 20th, but I haven't read anything about the release of Samsung's information. Does anybody know more?

3 ^ | v • Reply • Share ›

Christian Affolter • 3 years ago

Any news from Samsung already? I'm keenly awaiting your next update, as we also experience similar issues within parts of our own infrastructure, running the same SSDs. Until now, we do not know if it's the RAID controller or the SSD which is to blame. Thanks for keeping us posted.

2 ^ | v • Reply • Share ›

Adam Surak ➚ Christian Affolter • 3 years ago

Just posted an update. No progress so far. The only way around is to disable any usage of trim in the meanwhile. You experience the same issues?

^ | v • Reply • Share ›

Heinz Kurtz ➚ Christian Affolter • 3 years ago

Which raid controller(s) are you using?

^ | v • Reply • Share ›

Christian Affolter ➚ Heinz Kurtz • 3 years ago

The affected nodes are using either Adaptec 7805 or 72405

^ | v • Reply • Share ›

anon • 3 years ago

BTW, this also sounds similar to the recent mdraid bug with raid0 that

BTW, this also sounds similar to the recent mdraid bug with raid0 that was introduced in various stable kernels, where TRIM would actually trim the wrong files/regions, zero-ing out random files on the file system. Seems like TRIM is pretty dangerous. :-) See: https://lkml.org/lkml/2015/...

2 ^ v • Reply • Share ›

Adam Surak ➚ anon • 3 years ago

This got introduced in 4.0 afaik. We thought it might be the raid0 but since it did not happen on different drives with identical software stack, we had to go lower.

^ v • Reply • Share ›

Heinz Kurtz • 3 years ago

NCQ causes a lot of issues with a lot of drives and it wouldn't be the first time Linux blamed drive firmware for its own bugs. I've not seen any performance issues since disabling NCQ globally so I rather not risk it. There should be a stress test tool that does random read/write/delete/trim and verifies integrity. Something like a badblocks for trim, that you could run on each new setup for a day or so before going productive.

The problem with TRIM is that it's literally everywhere. Enabled by default and performed unasked by mkfs and other tools. Some people set issue_discards in lvm.conf because they misunderstand that as allowing trim instead of having LVM itself perform wantonly trim on each lvremove lvresize snapshot etc.

If possible do TRIM in a cron-job using fstrim or such. If things blow up right after the cron-job ran it's easier to pinpoint the cause of the trouble.

2 ^ v • Reply • Share ›

Heinz Kurtz ➚ Heinz Kurtz • 3 years ago

Oh and if you want to run your own tests, beware: Linux cheats. It does not remove from its caches what has been trimmed. So the only way to verify that data is still intact after TRIM is to drop caches to force re-read from hardware (and hope the hardware does not have its own cache).

2 ^ v • Reply • Share ›

Stu • 3 years ago

I've had weird read errors from time to time on my crucial m4 SSD as well. ... When doing lots of reads (like trying to backup), and even randomly returning wrong data from files (also when i had done a lot of reads and writes building Python virtualenvs).

Unfortunately I'm traveling so can't order a replacement of a different make right now, and it seems to work about 99% of the time, those other times are worrying though.

2 ^ v • Reply • Share ›

CustomDesigned ➚ Stu • 3 years ago

Rotating disks store the sector number with each sector. Planning the motion of the R/W head to grab a sector is a complex realtime operation - and sometimes it get the wrong one. But it knows that immediately, and can try again, doing a seek recalibrate if needed. A momentary freeze is FAR better than silent data corruption.

You would think SSD manufacturers would do the same. If you ask the drive for sector 123456, and the FTL (Flash Translation Layer) points to sector 7654321, the controller can do a sequential search for the correct sector and fix the corrupted FTL. A freeze during that process if FAR better than silent data corruption.

^ v • Reply • Share ›

Santiago Franco • 3 years ago

Hi, just bought a new Samsung SSD 850 EVO (Not Pro), and installed Ubuntu 15.10. After reading this, (I had some strange index errors after a couple of days with everything fine). I disabled Trim and now it works, (login takes 5 mins approx, which should be much faster with a SSD), at least no error messages on booting. Does anybody know if this is fixed, and if not, should the SSD drive be ok without TRIMs or will performance degrade with use? Thanks!

1 ^ v • Reply • Share ›

Gerhard Islinger • 3 years ago

While all seems to center on Samsung and it being a Linux bug... In that case i find it puzzling that it works with Intel SSDs...? Could anyone test it on Intel?

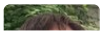
Or am i missing something?

1 ^ v • Reply • Share ›

Heinz Kurtz ➚ Gerhard Islinger • 3 years ago

We are still missing the patch itself, it should solve the mystery. Bugs can be weird, and trigger only under obscure circumstances, if it happens only under specific loads it could be a timing issue or whatever.

1 ^ v • Reply • Share ›



Anders Aagaard • 3 years ago

Checking in on this every now and again, really appriciate you guys trying to get to the root of the problem, and updating the blog along