The Old New Thing

How long does it take to create a 16TB file?

4 Dec 2007 10:00 AM 56

Although the theoretical maximum file size on NTFS is 2^{64}−1 clusters, the current implementation of the NTFS driver supports files up to "only" 16TB minus 64KB. (In other words, the disk format supports files up to 2^{64}−1 clusters, but the current drivers won't go above 16TB−64KB.)

Back in 2002, in order to verify that the drivers did indeed support files as big as their design maximum, the NTFS test team sat down, created a volume slightly bigger than 16TB, and then created a file of the maximum supported size, filling it with a known pattern. After the file was successfully created, they then ran another program that read the entire file back into memory and verified that the contents were correct. (They ran other tests, too, of course, but those are the ones that are important to this story.)

How long did it take to create this nearly-16TB file?

Around three days.

Verifying that the data was written correctly took about four days.

(Yes, it's strange that reading was slower than writing. I don't know why, but I can guess and so can you. Maybe the read test did a bunch of extra verification. Maybe the read test used random access as well as sequential access. Or maybe there was just rounding error in the reporting of the duration. I wasn't there, so I don't know for sure.)

Blog - Comment List MSDN TechNet

Comments

AB
4 Dec 2007 10:18 AM
#

IIRC, on NTFS, if you create a file and then set the EOF pointer to a huge value, you can create a huge file almost instantly.

However, there is a catch - once you write to any position in the file, all bytes before that position will be physically initialized, resulting in a long wait.

If you have SE_MANAGE_VOLUME_NAME privilege, you can skip the zero-initialization altogether by calling SetFileValidData() API.

This way, the file creation will be truly instant, and the file will contain whatever was written in those clusters before the file was created.

(also see FSCTL_SET_ZERO_DATA ioctl code)

Dataland

4 Dec 2007 10:39 AM
#

AB,

As you have mentioned, a large file can be created much faster, but doing so does not exercise much of the File IO APIs. I mean, if you are going to test that a 16TB can be properly created, you will really want to write the whole dang thing. And then, read the whole dang thing, and while you are at it, you probably want to run a bunch of other File APIs to make sure that the 16TB file works just like a 16KB one.   So, yep there are shortcuts but these (at least IMHO) defeat the purpose of what they were trying to do.

Bobaloo
4 Dec 2007 10:43 AM
#

You know, I've finally got to ask: What the heck is this stupid coderblogs thing that keeps cropping up in blog comments? Are your posts being duplicated with or without your permission? If without, why isn't someone at Microsoft threatening legal action and/or sending DMCA takedowns on their employee's behalf?

This kind of stuff really burns my butt if it's done without permission, and if it is with permission it really should say so somewhere on this guy's site.

Michael Dwyer
4 Dec 2007 11:43 AM
#

See? Microsoft /does/ do testing!

Not that I had any doubts, but it somehow really pleases me to see that somebody's burning a week to check one edge case. That's somehow really cool and really disturbing at the same time...

Serge Wautier
4 Dec 2007 12:42 PM
#

Forgive my ignorance about storage in general and NTFS in particular but I guess they had to spread the volume accross several physical disks, right? How does one do that? Or did they have a 16TB disk back in 2002?

Just curious...

[*I understand there are these things called "search engines" that are pretty good at helping you look up up product features. -Raymond*]

Mihai

4 Dec 2007 12:42 PM
#

Nitpicker's corner: with the following statement "read the entire file back into memory" does not mean that the full file was present in memory at one time (which requires more than 16TB of memory).

Ok, fixed that :-D

pplu
4 Dec 2007 12:44 PM
#

Tests should be done on edge cases (that's one of the points in good testing methodologies).

Why is it disturbing? It's not like it's going to make windows lauch a week later... and if it did... I prefer a late launch with no NTFS file system flaws to an update because files beyond x size are not properly handled.

By the way, Raymond, and just out of curiousity: on the fundamental pieces of the OS (like NTFS) what are the coverage metrics? (I hope they are disclosable, and a proud 100%)

Aaargh!
4 Dec 2007 12:59 PM
#

I wonder if anyone in the Real World(tm) is actually using NTFS to store huge files like that ? Wouldn't you use a more suitable filesystem ? Is it e.g. even possible to resize a live NTFS filesystem ?

What alternatives are there on windows beside NTFS and FAT32 ?

Gabe
4 Dec 2007 1:14 PM
#

Writing a 16TB file in 3 days means a constant 60+ MB/sec, or a 64kB transaction every millisecond.

Does anybody know how many disks they used? I would have to imagine it's in the hundreds! The nice thing about a test like this is that redundancy is unimportant, so you could make a huge RAID 0 stripe set.

John
4 Dec 2007 1:15 PM
#

But how long does it take to copy a 16TB file?

Aaargh!
4 Dec 2007 1:28 PM
#

"Forgive my ignorance about storage in general and NTFS in particular but I guess they had to spread the volume accross several physical disks, right? How does one do that?"

Either some kind of RAID or Logical Volume Management. Basically, you have some software (or hardware) that maps the logical adress space of the volume to the different physical disks.

Dataland
4 Dec 2007 1:48 PM
#

Personally, I think it's great that MS sweats the edge cases! I also think it would be great for PR, and edu-tainment if more "inside info" like this was blogged.

On a related topic... Over the past decade I've been disappointed that MSDN magazine has replaced stories/sagas/inside-scoops (i.e. Raymond-type articles) with developer-focused-marketing of Complexity.Stew.And.Cool.New.Technologies. I guess what i'm trying to say is I find Raymond's MSDN mag articles (and this blog) much more interesting/useful/helpful than the rest of MSDN mag.

anonymous
4 Dec 2007 2:12 PM
#

AB, you don't need this privilege

to use FileSetValidData(). Anyway, you could simply create a sparse file and set the EOF pointer to a large value. Again, without any special privileges.

Maybe you're confused because the FSUTIL command utility doesn't run without administrator privileges. A quick patch making a JNZ into an unconditional JMP fixes this issue.

[*Nitpicker's corner: "... to create a 16TB file with meaningful contents, not uninitialized garbage." I can't believe I had to write that. -Raymond*]

sandman
4 Dec 2007 2:38 PM
#

Didn't online resizing for NTFS get added in the Vista kernel? (I could be wrong here - I haven't needed to do it yet).

But yes, all the uses I can think of for a 16TB filesystem (other than test like the story) I would want the enterprise features of online resize and Logival volume management.

sandman
4 Dec 2007 2:39 PM
#

Yeah, A sparse file won't help the test case - but why would it have uninitialised garbage - surely the kernel substuites zeros for the unallocated blocks?

[*I was referring to the original comment which suggested SetFileValidData. -Raymond*]

DriverDude
4 Dec 2007 2:40 PM
#

Back in 2002, the disks were 160 GB max (assuming cheap large ATA disks, as opposed to small fast SCSI) So they would've needed 100+ disks and some serious enterprise equipment just to make a 16 TB volume.

I'm impressed the NTFS team could requisition that kind of hardware just for a test. Obviously, I hope they took the time to run other tests too.

E.g., has anyone created 2^32 -1 files on a NTFS volume yet?

Aaargh!
4 Dec 2007 2:46 PM
#

"Didn't online resizing for NTFS get added in the Vista kernel? (I could be wrong here - I haven't needed to do it yet)."

I tried looking it up on MSDN but that's completely useless, can't find a single piece of usefull information. The info on wikipedia also seems out of date.

mh
4 Dec 2007 2:48 PM
#

It always amazes me the extent to which the MS folks go to test things. I could well imagine a lot of others (myself included) assuming that nobody would ever want a 16 TB file, and just not bothering.

This one is right up there with "the USB cart of doom" in my list of personal favourites.

thomas.tmc
4 Dec 2007 2:56 PM
#

As far as getting the needed hardware goes, MS spends several billion dollars a year in research. They probably used leftover components from other research projects, ie a 3TB array here, a 5TB array from there, and pieced them together.

It wouldn't surprise me if you could walk into a storeroom at an MS research building and find a box with 100 160GB drives and 40 barbones boxes just sitting in shrink wrap.

David Walker
4 Dec 2007 3:48 PM
#

I'm NOT impressed that the test team could requisition 100 disks "just" for such a test. It seems like a fundamental thing to test.

If you claim that your OS can create a 16 TB file, you ought to test it, and "the" test team certainly ought to be given the resources to do this. MS could easily have afforded this in 2002!

I'll bet MS had tons of enterprise-class equipment even back in 2002.

Mark Steward
4 Dec 2007 4:09 PM
#

Online resizing is available in Vista:

http://windowshelp.microsoft.com/Windows/en-US/Help/f2e9a502-e63c-413d-8804-87326ef4f4cc1033.mspx

Rob
4 Dec 2007 4:53 PM
#

Diskpart was introduced around the Windows 2000 SP1 time frame. Diskpart will allow you to extend a data drive on the fly. See KB325590
http://support.microsoft.com/default.aspx?scid=kb;EN-US;325590

Vista will let you extend or shrink volumes including the OS drive.

Cooney
4 Dec 2007 4:54 PM
#

Any idea about XP? I don't really want to run Vista. Ever.

Evan
4 Dec 2007 5:11 PM
#

pplu: "By the way, Raymond, and just out of curiousity: on the fundamental pieces of the OS (like NTFS) what are the coverage metrics? (I hope they are disclosable, and a proud 100%)"

100% coverage for just about any well-written software, both in practice and in theory.

Evan
4 Dec 2007 5:20 PM
#

Hmmm, I probably should have finished writing my previous post before posting. It should have read:

"100% coverage IS IMPOSSIBLE for just about any well-written software, both in practice and in theory."

This is true even if you're talking statement coverage. If you talking something like path coverage, it becomes even worse.

Gabe
4 Dec 2007 5:36 PM
#

I'm pretty sure that NTFS volumes have always been able to be extended online. The partition generally could not be expanded, though. You would just create a new partition (possibly adjacent to the old one), and expand the volume into the new partition. The space would immediately become available.

Nar
4 Dec 2007 5:37 PM
#

>"100% coverage IS IMPOSSIBLE for just about any well-written software, both in practice and in theory."

>This is true even if you're talking statement coverage.

How? Unit testing can get every method covered easily, and if your methods are so convoluted that it's hard...

James Day

4 Dec 2007 5:38 PM
#

If the information in the encyclopedia hosted at Wikipedia.org is out of date, there's a simple solution: fix it. That's why it's a wiki.

Remember to include links to your sources so what you change can be verified by others.

mh
4 Dec 2007 5:58 PM
#

"Any idea about XP? I don't really want to run Vista. Ever."

Diskpart will work very happily with XP too. You can get even more flexibility by using dynamic disks. Plenty of documentation out there if you're interested.

The main limitation comes in if you want to shrink a partition. You're also restricted from extending the system or boot partition(s). But if you're creative enough you can work around that with the tools you get for free (a combination of WinPE, ImageX and a spare partition springs to mind).

DriverDude
4 Dec 2007 6:00 PM
#

Back-of-envelope says $10K of hardware today - 17 disks, couple of RAID cards and one computer - can do this same test (write and read) in less than two days. Dell or Apple Xserve boxes cost more but it still won't break the bank or table.

I wonder if these tests are in the NTFS regression test suite...

Evan
4 Dec 2007 6:04 PM
#

"How? Unit testing can get every method covered easily, and if your methods are so convoluted that it's hard..."

At the very least, what about stuff like

switch (x) {

case A:

  ...

case B:

  ...

case C:

  ...

default:

  // this should never happen

  return EINVAL;

}

You also have the condition of asserts, which should never fire, and it's not a bad idea to keep stuff like that around for a number of reasons.

Legolas
4 Dec 2007 6:23 PM
#

I wonder how fast that test would be done today. And I'm slightly surprised it would still take over a dozen disks today... So did they code and test a larger one yet? (Ok perhaps you can't talk about that sort of thing yet. Would be a cool story though ;-)

sandman
4 Dec 2007 6:45 PM
#

MH: " You're also restricted from extending the system or boot partition(s)."

That's characteristic though of offline resizing where the volume cannot be dismounted. The question was about online where a partition is resized while filehandles are open on the volume. I'm not sure that Mark's link clears the matter up either.

nksingh
4 Dec 2007 7:55 PM
#

I've definitely recently extended an NTFS partition on my laptop to use space from a defunct Linux partition. This worked as expected on Vista.

I think I also had shrunk the partition before when I first installed Linux, but I'm not as sure about that.

Testing limits is hard when the limits are chosen to be bigger than currently available hardware. And they can't just set a #define somewhere because the file system and memory manager work hand in hand and they'd definitely want to track down non-obvious interactions between limits in both of them.

I bet the most costly part of writing that huge file was multiple traversals of the MFT File Record and perhaps the creation of multiple linked overflow records.

microbe
4 Dec 2007 8:09 PM
#

Reading is slower because it's mostly synchronous (there probably wasn't a very huge buffer and the prefetch probably wasn't aggressive enough), while writes are asynchronous.

Mark Steward
4 Dec 2007 8:14 PM
#

sandman: I just shrunk my only drive while running Vista by 50MB and then extended it again. Just for you ;-)

You're restricted from extending the system or boot partitions only on XP and its ilk.

GreaseMonkey
4 Dec 2007 8:15 PM
#

For 16TB, you could rip nearly 1,800 dual-layer DVDs, uncompressed. Anyone who would have that many DVDs is either a crazy movie fanatic, or on the run from the RIAA.

If you're talking about the full 16EB, though, that'd be 1,800,000,000 DL-DVDs. And I guess it'd take many years to fill *THAT* much space. Heck, after pirating that much, you could be out of prison by the time it's finished.

Simon Cooke
4 Dec 2007 8:42 PM
#

I find it a fun exercise to come up with the pattern writer that would be used to write the data out to the file...

After all, you're not going to just want sequential access tested, you're going to want random access tested too, so you can't use a linear congruential random number generator. (Unless you partition it into chunks, and use the chunk # as the seed for the start of the chunk).

It's times like this that I really like the idea of writing out the digits of PI and filling that space up using the Bailey-Borwein-Plouffe algorithm.

You get nice random-ish data, recoverable in constant time, for any point in the file. :)

DriverDude

4 Dec 2007 8:58 PM
#

"I bet the most costly part of writing that huge file was multiple traversals of the MFT File Record and perhaps the creation of multiple linked overflow records."

I thought the metadata would be cached and only flushed to disk a few times...

Also, why would multiple overflow records be needed, assuming most of the file is contiguous?

On the other hand, how many non-contigous extents can NTFS deal with, if there were a few million files already scattered across the volume?

Corner cases involving large numbers are interesting indeed.

Gazpacho
5 Dec 2007 12:00 AM
#

Any organization with 16TB of data sitting around would certainly be big enough to put the hurt on Microsoft if it had failed to test the file system.

Oh, and hello Simon. Haven't seen you since back in the day on MSNBC.

sandman
5 Dec 2007 2:18 AM
#

DriverDude:"I thought the metadata would be cached and only flushed to disk a few times..."

Maybe, but NTFS is metadata journalled.

You must write the metadata fully out to the drive when you complete a checkpoint. And if you are extending a 16TB file there's a lot of metadata. How many checkpoints where needed is an interesting question.

Interestingly - in some ways it would be safe not to write the metadata from file extension to the journal - as the lack of a metadata write is an automatic rollback. Without FileSetValidData() the data _must_ be checkpointed before or simulatanosuly with the metadata.

Of course since you realistically need to ensure metadata writes are ordered and shrinking, or deleting a files metadata writes must happen before that files block is reallocated. (Otherwise there is a security whole if you can force an system crash and reload at the right moment). This implies it is simpler just to journal all metadata writes....

The upshot of this is there could be quite a lot of metadata activity as the file is allocated,

Neil
5 Dec 2007 5:55 AM
#

>But how long does it take to copy a 16TB file?

If you're using the same hardware and APIs as the NTFS test team, then I'd say about six to seven days. (I say that because I believe that CopyFile would not be suitable in this case.)

Danny
5 Dec 2007 10:23 AM
#

>Tuesday, December 04, 2007 2:40 PM by >DriverDude

>E.g., has anyone created 2^32 -1 files on a >NTFS volume yet?

I did that last year when I got hands on a DVD with Fable game and put Nero to do a image - at work. Was 4GB file. And at home I had FAT32, so image my face when I finally realized is time for converting my FAT32 to NTFS :)

David Walker
5 Dec 2007 10:26 AM
#

You don't need to close ALL open handles on a volume in order to extend or shrink it.

You might need to delay some new file creations or deletions, or new allocations in the bitmap, while you rewrite some things, but the currently open files ought to be able to stay open.

I don't know why Windows 2000 did not provide a native way to resize the system or boot partition... I suppose it's just a case of not allocating the resources to write the software, which is perfectly understandable.

The mainframe operating system TPF (Transaction Processing Facility, formerly known as ACP for Airline Control Program), which can typically handle 5000 transactions per second or more, has always had programmatic support in the operating system with on-the-fly ability to add (or remove) entire disk drives to its storage array.

It then proceeds to "level" the data across all of the disk drives, by moving data records from the fuller disks to the new disks, while still processing thousands of transactions per second.

Again, I suppose it's a matter of prioritizing programming resources. But don't assume that something like that can't be done.

(As a related example, critical system files COULD be replaced in memory while the operating system is running. Consider the state of memory and disk (system folders) before the operation, and the state of memory and disk that you would get after file replacement and a regular reboot, and figure out a way to map everything from one to the other, and implement that mapping in software. Some locks would be required. It's hard, but not impossible.)

Ryan
5 Dec 2007 11:38 AM
#

My organization has more than 16TB of data online, and we have just 120 employees and under $100M in revenue. Most if this is SQL Server database files.

We're not big enough to put the hurt on MSFT for anything.

anomymous
5 Dec 2007 12:44 PM
#

A question: Why did their use some static patterns instead of random data? Just create a cryptographic checksum of the data while writing them, then read it back while checksumming it, and compare the checksums.

DriverDude
5 Dec 2007 1:27 PM
#

"Just create a cryptographic checksum of the data while writing them, then read it back while checksumming it, and compare the checksums."

That tells you whether the file changed, but doesn't indicate what or where. A known pattern can indicate whether a portion of the file was written over itself, etc.

Of course, that assumes you pick a useful pattern, one with a unique identifier every 512 bytes (like a block counter and timestamp)

Gazpacho
5 Dec 2007 2:05 PM
#

If you hash 16TB with a 20-byte hashing algorithm, then you can expect a lot of other bit patterns to hash to the same value. If the hash is uniform, there will be more than 800G such bit patterns, which may or may not be significantly different from each other. In short, I would be at a loss to give a manager any reason for doing it that way instead of the direct way.

However, if it was me doing the test I would fill the file with data from a cryptographic PRNG, and then verify it with the same seed value.

Gabe

5 Dec 2007 4:02 PM
#

Danny, he meant a volume with 2^32-1 files, not a file with 2^32-1 bytes. Since each file will require at least a 1kB record in the MFT, you would need at the very minimum a 4TB volume to hold that many files. You would also need some additional space for overhead (volume bitmap, an index of those 4 billion files, etc.). Anyway, the volume would have to be 4.5e12 bytes or so.

joe
5 Dec 2007 6:07 PM
#

16 TB? Was that 1 TB = 1,000 GB, 1 GB = 1,000 KB, 1 KB = 1,000 bytes like the drive manufacturers use?

Chris Marshall
5 Dec 2007 6:29 PM
#

Doesn't nearly every CD/DVD drive produced in the last 5 years write faster than it reads?

Stefan Kanthak
5 Dec 2007 8:08 PM
#

Big SCSI disks are a myth!

Back in 2001 I got a bunch of small fast SCSI disks for my RAID @home: Seagate ST118677LC, each 180GB small.

Gabe
5 Dec 2007 11:11 PM
#

joe: That file was 17,592,185,978,880 bytes. It's 268,435,455 (2^28-1) clusters, each 65,536 (2^16) bytes. Or as it says in the article Raymond linked to, it's 2^44 bytes - 2^16 bytes.

In SI units, I guess that would be 16TiB or 17.6TB. It would require 18 disk drives rated to hold 1TB each.

anomymous
7 Dec 2007 9:24 AM

#

"That tells you whether the file changed, but doesn't indicate what or where."

Well, since we're only making a go/no-go test we only care if the file was changed or not. If everything is OK then there should be no change, and since this is the most likely case, we can optimize for it.

[*Good luck with that random access testing. -Raymond*]

-
7 Dec 2007 11:59 AM
#

> the theoretical maximum file size on NTFS is 2^64−1 clusters

Haha. No. Not even close. Try again with 2^64−1 bytes.

Gabe
8 Dec 2007 3:15 AM
#

If you believe the article, neither 2^64-1 clusters nor 2^64-1 bytes is correct. It's 2^64-1024 bytes. To quote the article, "Theory: 16 exabytes minus 1 KB (2^64 bytes minus 1 KB)".