**[GNU Parted: Solving the dreaded "The resulting partition is not properly aligned for best performance"](https://blog.hqcodeshop.fi/archives/273-GNU-Parted-Solving-the-dreaded-The-resulting-partition-is-not-properly-aligned-for-best-performance.html)**

**Saturday, November 7. 2015**

On the other day I was cleaning out junk from my shelfs and found a perfectly good [**WD Caviar Black**](http://www.wdc.com/en/products/products.aspx?id=760) hard drive. Obviously in the current SSD-era where your only computer is a laptop and most of your data is stashed into a cloud somewhere, no regular Joe User is using spinning platters.

Hey! I'm not a regular, nor joe. I have a Linux-server running with plenty of capacity in it for my various computing needs. So, the natrural thing to do is to pop out one of the old drives and hook this 1,5 TiB high performing storage monster to replace it. The actual hardware installation on an ATX-case isn't anything worth documenting, but what happens afterwards goes pretty much this sequence: 1) partition the drive, 2) copy all/some of the old data back to it and 3) continue living successfully ever after.

The typical scenario is that something always at least hiccups, if not fails. And as expected, I choked on the 1).

Here goes:

**Preparation**

The drive had been used previously, and I just wasted the beginning of the drive by writing 10k sectors of nothingness. This will remove all traces of possible partition tables, boot sectors and all the critical metadata of the drive you normally value highly:

**# dd if=/dev/zero of=/dev/sdx bs=512 count=10000**

Pay attetion to the details. It would be advisable to target a correct drive. In my case a regular [**JBOD**](https://en.wikipedia.org/wiki/Non-RAID_drive_architectures#JBOD)-drive really appears as /dev/sda on the Linux-side. On your case, I'm pretty sure your operating system runs on /dev/sda, so please don't wipe that.

Then with [**GNU Parted**](http://www.gnu.org/software/parted/), create a [**GUID partition table**](https://en.wikipedia.org/wiki/GUID_Partition_Table) (or GPT):

# parted /dev/sda  
GNU Parted 3.1  
Using /dev/sda  
Welcome to GNU Parted! Type 'help' to view a list of commands.  
(parted) **mktable gpt**

That's it for the preparation part.

**Attempt 1: The stupid way**

Regardless what's on the drive already (in my case, its completely empty), Parted syntax allows an approach, where you create a partition using the maximum allowed capacity from start 0, to end -1. Like this:

(parted) **mkpart LVM ext4 0 -1**  
Warning: The resulting partition is not properly aligned for best performance.  
Ignore/Cancel? **c**

That obviously will emit an error about non-optimal partition alignment. But hey, that's what I asked for. I obviously cancelled that attempt.

**Attempt 2: The smart way**

A smart approach would be to see about the boundaries:

(parted) **print free**  
Model: ATA WDC WD1502FAEX-0 (scsi)  
Disk /dev/sda: 1500GB  
Sector size (logical/physical): 512B/512B  
Partition Table: gpt Disk Flags:  
  
Number Start  End    Size   File system Name Flags  
       17.4kB 1500GB 1500GB Free Space

Now we have a range of 17.4 KiB to 1500 GiB which can be used for a new partition. Let's try that:

(parted) **mkpart LVM ext4**  
Start? **17.4kB**  
End? **1500GB**  
Warning: You requested a partition from 16.9kB to 1500GB (sectors 33..2929687500).  
The closest location we can manage is 17.4kB to 1500GB (sectors 34..2930277134).  
Is this still acceptable to you? Yes/No? **y**  
Warning: The resulting partition is not properly aligned for best performance.  
Ignore/Cancel? **C**

I have bumped into this number of times earlier. Why in the f\*\*k cannot the Parted tell me what values it wants to see there!! Come on! :-(

This is the part where it hits me like a hammer: enough bullshit, let's solve this once and for all!

**Attempt 3: Solution**

This is the script I wrote: [**parted\_mkpart\_calc.sh**](http://opensource.hqcodeshop.com/Parted%20calculator/parted_mkpart_calc.sh).

It is based on the information found from following sources:

* [**How to align partitions for best performance using parted**](http://rainbow.chard.org/2013/01/30/how-to-align-partitions-for-best-performance-using-parted/), somebody else is having the same fight than I do
* [**I/O Limits: block sizes, alignment and I/O hints**](http://people.redhat.com/msnitzer/docs/io-limits.txt), information about the Parted alignment calculation
* [**https://www.kernel.org/doc/Documentation/ABI/testing/sysfs-block**](https://www.kernel.org/doc/Documentation/ABI/testing/sysfs-block), Linux kernel block-device ABI information

It is a [**Bash**](https://www.gnu.org/software/bash/)-script to do the math for you. Example usage:

$ **./parted\_mkpart\_calc.sh sda**  
Using default 1 MiB default alignment in calc  
Calculated alignment for /dev/sda (gpt) is: 2048s  
  
If you would be root, you could create partition with:  
# parted /dev/sda mkpart [name] [type] 2048s 2930276351s  
Verify partition alignment with:  
# parted /dev/sda align-check optimal 1 Should return: 1 aligned

I just enter one argument to the script: sda. From that, the script deduces the alignment, that should be used when partitioning that block-device. In this case it is 2048 sector boundaries (what it doesn't say is, that a sector contains 512 bytes). But it outputs 2 commands which can be copy/pasted (as *root*):

parted /dev/sda mkpart [name] [type] 2048s 2930276351s

If you would replace [name] with a partition name and [type] with a partition type, it would create a correctly aligned partition to fill up most of the drive. It won't fill up exactly all of the drive, because of the alignment issues.

To help that issue, I added a feature to do the following:

$ ./parted\_mkpart\_calc.sh sda LVM ext4

Optionally, you can provide the partition name and type on the command line to get:

parted /dev/sda mkpart LVM ext4 2048s 2930276351s

as output. That's ready-to-go copy/paste material.

Finally, you can verify the correct alignment:

# **parted /dev/sda align-check optimal 1**  
1 aligned

That's the proof, that calc worked ok.

**Attempt 4: The simple way**

It didn't take long, before I got my first comment on this article. It was simply: "*Why didn't you use percentages?*". What? What percentages.

Example:

(parted) **unit s**  
(parted) **print**  
Model: ATA WDC WD1502FAEX-0 (scsi)  
Disk /dev/sda: 2930277168s  
Sector size (logical/physical): 512B/512B  
Partition Table: gpt  
Disk Flags:  
  
Number Start End         Size        File system Name Flags  
 1     2048s 2930276351s 2930274305s             LVM  
(parted) **rm 1**  
(parted) **mkpart LVM ext4 0% 100%**  
(parted) **print**  
Model: ATA WDC WD1502FAEX-0 (scsi)  
Disk /dev/sda: 2930277168s  
Sector size (logical/physical): 512B/512B  
Partition Table: gpt  
Number Start End         Size        File system Name Flags  
 1     2048s 2930276351s 2930274305s             LVM

Using range 0% 100% will produce exactly the same results. :-) Amazing!

So, parted knows the alignment and can use it, but not if you don't first do a rain dance and knock three times on a surface sprinkled with holy water.

**Final Words**

Why does Parted complain about mis-alignment, but offers no help at all? That's just plain stupid!

Of course, I should add the feature to the source code and offer the patch to [**FSF**](http://www.fsf.org/), but on the other hand. Naah. I don't want to waste any more energy on this madness.