Advanced - Creating aligned rpool partitions

It may be possible that administrative problems similar to those described in the article Larger sector size drives (4K and "Advanced Format") can happen with other hardware – for example, some SSDs claimed 8K sectors and manipulation of 256K-512K pages for reprogramming of their "Flash"-chips, or SAN LUNs (including those backed by ZFS volumes over iSCSI or FC) can operate most efficiently with blocks of a specific size. Misalignment may place an unnecessary toll on IO performance (by regularly having logical blocks contained across different sectors and requiring more IO operations and bandwidth to access, as well as requiring RMW operations – and delays – to update), and possibly on storage reliability (more wear on SSD devices due to more required IO, and in particular misaligned ZFS pool labels might get "torn" across too many hardware sectors and pages).

Good alignment usually means that the absolute starting offset from the start of disk of the `s0` slice (of Solaris disk label) dedicated to the `rpool`, encapsulated in its MBR partition, should be divisible by the hardware sector size, and size of this slice should be a multiple of the "cylinder size" (as defined by Solaris `format` for this disk's type) ***and*** of the hardware sector size as well.

> ⚠ **NOTE: Speculation and guesswork in the paragraph below**
>
> Depending on the (generally unknown) implementation of the storage controller on SSD devices, it *may seem* (I am not really sure about *"must be"*) desirable to configure your `rpool` device, and possibly other partitions on this storage device, such as for ZIL and L2ARC usage for the data pool in smaller setups like a SOHO NAS server, in such a manner that IO's and storage of important system components (such as the pool labels) are well-aligned considering not only the sector size, but also the page size. In case of SSDs with a page size of 256Kb it seems natural to desire that the four ZFS label copies are 256K-aligned in terms of physical offsets. Possibly, for 512K pages steps should be taken to assure that the four labels fall into different pages as well (playing with odd and even numbering in 256K alignment). Although, arguably, wear-leveling in modern SSD firmware and colocation of different LBAs in the same hardware pages may negate this particular effort.

Technical nuances

First of all, let's go over the technical basics – what do these layouts look like as offsets of the on-disk allocation?

MBR	SLICE	Description
`p0`		LBA 0, Physical offset 0 (rarely 1) – Start of MBR (the MBR table itself is in sector number 0, along with the initial bootloader code)
`p1`	`s2`	Start of MBR partition 1 (which in our example contains the Solaris SMI labels and the `rpool` slice) The slice #2 `backup` traditionally addresses the whole partition from start to end
	`s8`	`boot` slice (#8), one cylinder in size
	`s9`	`alternates` slice (#9), optional as decided by the OS, two cylinders in size
	`s0`	The slice #0 for `rpool` component (single-disk or mirror-half) – this is the point which should be well-aligned
	`s0'`	The end of slice #0 for `rpool` which should also be well-aligned
	`s1...`	Optional other slices in this label (maybe for ZIL or L2ARC devices, or components of other pools – usable by Solaris/illumos only)
		2 "reserved" cylinders (optional as decided by the OS?)
`p2...`		Optional other partitions for other OSes (dual-booted systems), or for ZIL (`log`) or L2ARC (`cache`) components for other (non-root) pools on small systems

NOTES:

- The OS somehow decides about the "disk type" to assign to the storage device, and this influences both the size of "cylinders" (most commonly 63*255=16065 "blocks" of 512 bytes each) and the presence and size of the "alternates" slice. The disk type can be reviewed and may be changed in the `format` command interface. The slice sizes are multiples of cylinder size.
  This may be a problem when replacing the `rpool` disks' connection type (i.e. from Legacy IDE to Native SATA) or extending a mirror, which might cause two identical drives to be assigned different "types" and, in my experience, different cylinder sizes (so the SMI table could not be copied from one disk to another, because the slice sizes are counted in full cylinders). Setting the new disk's "type" to that of the old disk did fix the problem for me.
- To complicate matters worse, the MBR partitions "should" start at track-sized offsets (63*512 bytes) to keep legacy tools content, so you might care if you build a multibooting system, or use a ghost-like low-level backup tool. Preferably, they should also fill a track to an end, leaving no gaps.
- As detailed in Larger sector size drives (4K and "Advanced Format"), some Advanced Format drives may have a setting to shift the logical addressing by one legacy sector size in comparison to hardware layout. This is done for better automatic compatibility with legacy OSes like Windows, which would partition the disk with `p1` starting at LBA #63 and would use "clusters" of 4K or more; with a one-sector shift such partitions would start at a hardware offset of 64 "legacy sectors", which is a multiple of these disk's hardware sector size.
- The table above slightly touches on the possibility of a disk being used by several pools (a root pool and a separate data pool for example), which may be substantial on systems with a constrained amount of disks – for example, I've managed a few servers with 4*HDD slots, using about 8-16Gb on each disk for a part of mirrored `rpool` or a mirrored `swap`/`dump` pool or even "raw device(s)", and the rest of the disks used as a 4-device `raidz1`, `raidz2` or `raid10` pool. If dual-booting is not a requirement, this is all best orchestrated by a single MBR partition spanning the whole disk, and a Solaris SMI slice-table inside it. On the other hand, components of non-root pools (including data or cache/log leaf VDEVs) are not required to be SMI slices, and may be MBR or GPT partitions...
- Take care to not use all of the disk, but leave several megabytes of slack space at the end, to simplify disk replacements in the future – the same marketed disk size may be a few sectors smaller than your old disk, and that would be a problem for replacements.

- Finally, it is important to know that the best practices urge against co-locating the `rpool` and L2ARC or ZIL devices, at least partially due to wear that the frequently written cache devices would induce on the media which also houses your operating system. If possible, store the `rpool` and the caches on different mirrored pairs, at least on production systems.

As detailed in the ZFS On-Disk Specification (page 7), the VDEV layout (content of the `rpool` slice) starts and ends with copies of the device labels:

0...256K	256K...512K	...data...	N-512K...N-256K	N-256K...N
L0	L1	...	L2	L3

The `L0` - `L3` are copies of the labels which are atomically updated in several separate IOs (`L0`, `L2`, `L1`, `L3`), so that at least one is probably intact in case of failures during a label update. The second half (128K) of each label contains a "ring of slots" for "Uberblocks", and the hardware sector size (or rather the a `shift` value) determines how many UB's fit there and thus how many rollback transactions may be possible – from 128 slots for 512b/512e devices down to 32 slots for 4k devices.

The hands-on math

Now, there is some math to do

The trick of this setup is that the slice used for `rpool` should start and end at the hardware-sector aligned offset, or even a page-aligned offset for SSDs; however the slice size should be a multiple of the "cylinder size" used by the OS for a particular disk.

First, you should create an MBR partition (with `parted` or `fdisk`, including an `fdisk` invocation from `format`) and a slice table in it (with `format`), just to see whether your OS instance would require the `boot` and/or `alternates` slices for the drive in question, and what cylinder sizes it would assign:

```
:; format
...
Specify disk (enter its number): 1
selecting c5d1
[disk formatted]
No Solaris fdisk partition found.

format> fdisk
No fdisk table exists. The default partition for the disk is:
  a 100% "SOLARIS System" partition
Type "y" to accept the default partition,  otherwise type "n" to edit the
  partition table.
y

format> p
partition> p
Current partition table (original):
Total disk cylinders available: 4099 + 2 (reserved cylinders)
Part      Tag    Flag     Cylinders        Size            Blocks
  0       root    wm       3 - 4098       31.37GB    (4096/0/0) 65802240
  1 unassigned    wm       0               0         (0/0/0)           0
  2     backup    wu       0 - 4098       31.40GB    (4099/0/0) 65850435
  3 unassigned    wm       0               0         (0/0/0)           0
  4 unassigned    wm       0               0         (0/0/0)           0
  5 unassigned    wm       0               0         (0/0/0)           0
  6 unassigned    wm       0               0         (0/0/0)           0
  7 unassigned    wm       0               0         (0/0/0)           0
  8       boot    wu       0 -    0        7.84MB    (1/0/0)       16065
  9 alternates    wm       1 -    2       15.69MB    (2/0/0)       32130
```

So, from the example partitioning above we can find that the system would assign the two reserved cylinders at the tail, as well as one overhead `boot` slice and two `alternates` slices, with the cylinder size being `16065` blocks (of 512 bytes each). *For the curious, this layout came from an SSD disk attached in LegacyIDE mode. Its twin, which arrived later and was attached straight to a SATA port, offered a very different layout (and disk type) in `format`. And the example is slightly edited – the actual disk was over 100Gb in size, but the partition for tests was manually created as a smaller one.*

Typical cylinder sizes are 16065 blocks and sometimes a value around 12 thousand, though on VirtualBox I often see 4096 blocks. The cylinder size can be verified in `fdisk` command as well (provide `p0` as the device); a VirtualBox example follows:

```
:; fdisk /dev/rdsk/c3t2d0p0
            Total disk size is 2520 cylinders
            Cylinder size is 4096 (512 byte) blocks
                                             Cylinders
      Partition   Status    Type          Start   End   Length    %
      =========   ======    ============  =====   ===   ======   ===
          1       Active    Solaris2          1  2519    2519    100
...
```

Continuing with the first example, let's assume that we use an SSD with 256KB pages (512 legacy sectors of 512 bytes) and 4Kb declared sector size (8 legacy sectors), and the disk is known or assumed to use a non-shifted absolute addressing (zero absolute offset is the real start of its hardware addressing for the purposes of alignment).

For some reason we do care about legacy partitioning tool compatibility and use track-sized offsets for the partition start and size (one track = 63 legacy sectors). And with all that, we want to hit the `s0` start aligned right on a multiple of 4096 bytes, and have the partition sized a whole amount of SSD pages).

p0	...	p1	0b	1a	2a	s0	...	s0'	1r	2r	p2	...
L>0 tracks = L*63 sectors		3 cylinders = 3*(63*255) sectors				S*cylinders*pages = S*(63*255)*512 sectors			1 cylinder = 63*255 sectors	1 cylinder = 63*255 sectors	...	...
N pages = N * 512 sectors												
M*tracks*pages = M * 63 * 512 sectors												

The table above, hopefully, describes all the values involved in this math. As a result of the desired layout, the starting offset and the size of `p1` partition will be a multiple of "track" size, while the start of the `p2` partition will also be SSD page-aligned and track aligned, with a minimal gap between the two. This keeps other partitioning tools happy, and IOs to other partitions should be efficient.

So, the absolute shift from start of disk to start of `s0` is a multiple of 256KB (or `N*512` legacy sectors), and it should be no less than one track plus 3 reserved cylinders (or `63*(1+3*255)`). The latter "lower bound" amounts to `63*766=48258` legacy sectors, or roughly 94.3 pages (of 256KB). This offset should also be a multiple of track size (63 sectors) because cylinder size incorporates tracks.

The nearest fitting solution for the offset of the slice is `p0..s0=1024*63` sectors.

In this case, the offset for the proper partition start is `p0..p1=(1024-3*255)*63=259*63=16317` sectors. This "wastes" roughly 8MB near the start of disk... let's consider it a small unexpected reserve of empty pages for SSD's redundancy rotation of chips 🤷

The slice size is a complicated beast – it should be a multiple of both the cylinder size and the SSD page size. In this example, the unit of measurement is `(63*255)*512` legacy sectors, or almost 4GB. A few of these are sufficient for a pool, depending on your installation size and expectations to expand (i.e. ability to add software and boot environments for safe updates, and whether you would store any `dump` or `swap` volumes on the `rpool`). In this example, I opted for an approximately 33GB `rpool`, or 8 such nameless size units.

Now we will wipe and recreate the partition table with proper values. Following is an example screenshot of a system actually configured with the math described above.

```
:; dd if=/dev/zero of=/dev/rdsk/c5d1p0 bs=512 count=1024

:; parted /dev/rdsk/c5d1p0
GNU Parted 1.8.8
Using /dev/dsk/c5d1p0
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) p
Error: /dev/dsk/c5d1p0: unrecognised disk label

(parted) mklabel msdos
(parted) pri
Model: Generic Ide (ide)
Disk /dev/dsk/c5d1p0: 120GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Number  Start  End  Size  Type  File system  Flags

### len=65882565 = 4101*16065 = 4096+3+2 cylinders
(parted) mkpart pri solaris 16317s 65898881s
(parted) toggle 1 boot

### On this SSD I also reserve ZIL and L2ARC:
(parted) mkpart pri 65899008s 76478975s
(parted) mkpart pri 76478976s 195826175s

(parted) uni s
(parted) p
Model: Generic Ide (ide)
Disk /dev/dsk/c5d1p0: 234440640s
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Number  Start      End         Size        Type     File system  Flags
 1      16317s     65898881s   65882565s   primary               boot  # rpool
 2      65899008s  76478975s   10579968s   primary                     # zil for data pool
 3      76478976s  195826175s  119347200s  primary   # l2arc stripe for data pool
     # 20Gb remains free for performance/longevity major boost on this SSD model
```

```
(parted) uni compact
(parted) p
Model: Generic Ide (ide)
Disk /dev/dsk/c5d1p0: 120GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Number  Start   End     Size    Type     File system  Flags
 1      8354kB  33.7GB  33.7GB  primary  solaris
 2      33.7GB  39.2GB  5417MB  primary
 3      39.2GB  100GB   61.1GB  primary
(parted) ^D
```

I hope the commands and outputs above are pretty self-explanatory 😓 The gap between `p1` and `p2` happens to be non-zero, but an acceptable whole two tracks (126 sectors).

Let's revise `format`'s suggestion of the Solaris SMI labelling, and create the slice for `rpool`:

```
# format c5d1
p
p
Current partition table (original):
Total disk cylinders available: 4099 + 2 (reserved cylinders)
Part      Tag    Flag     Cylinders        Size            Blocks
  0 unassigned    wm       0               0         (0/0/0)           0
  1 unassigned    wm       0               0         (0/0/0)           0
  2     backup    wu       0 - 4098       31.40GB    (4099/0/0) 65850435
  3 unassigned    wm       0               0         (0/0/0)           0
  4 unassigned    wm       0               0         (0/0/0)           0
  5 unassigned    wm       0               0         (0/0/0)           0
  6 unassigned    wm       0               0         (0/0/0)           0
  7 unassigned    wm       0               0         (0/0/0)           0
  8       boot    wu       0 -    0        7.84MB    (1/0/0)       16065
  9 alternates    wm       1 -    2       15.69MB    (2/0/0)       32130
partition> 0
Part      Tag    Flag     Cylinders        Size            Blocks
  0 unassigned    wm       0               0         (0/0/0)           0
Enter partition id tag[unassigned]: root
Enter partition permission flags[wm]:
Enter new starting cyl[3]:
Enter partition size[0b, 0c, 3e, 0.00mb, 0.00gb]: 4098e
partition> label
Ready to label disk, continue? y
partition> p
...
  0       root    wm       3 - 4098       31.38GB    (4096/0/0) 65802240
...
partition> ^D
```

Now we can proceed to Advanced - Creating an rpool manually.

HTH,
//Jim Klimov