NAME

virt−alignment−scan − Check alignment of virtual machine partitions

SYNOPSIS

```
virt−alignment−scan [−−options] −d domname

virt−alignment−scan [−−options] −a disk.img [−a disk.img ...]

virt−alignment−scan [−−options]
```

DESCRIPTION

When older operating systems install themselves, the partitioning tools place partitions at a sector misaligned with the underlying storage (commonly the first partition starts on sector `63`). Misaligned partitions can result in an operating system issuing more I/O than should be necessary.

The virt-alignment-scan tool checks the alignment of partitions in virtual machines and disk images and warns you if there are alignment problems.

Currently there is no virt tool for fixing alignment problems. You can only reinstall the guest operating system. The following NetApp document summarises the problem and possible solutions: http://media.netapp.com/documents/tr−3747.pdf

OUTPUT

To run this tool on a disk image directly, use the *−a* option:

```
$ virt−alignment−scan −a winxp.img
/dev/sda1        32256          512    bad (alignment < 4K)

$ virt−alignment−scan −a fedora16.img
/dev/sda1      1048576         1024K   ok
/dev/sda2      2097152         2048K   ok
/dev/sda3    526385152         2048K   ok
```

To run the tool on a guest known to libvirt, use the *−d* option and possibly the *−c* option:

```
# virt−alignment−scan −d RHEL5
/dev/sda1        32256          512    bad (alignment < 4K)
/dev/sda2    106928640          512    bad (alignment < 4K)

$ virt−alignment−scan −c qemu:///system −d Win7TwoDisks
/dev/sda1      1048576         1024K   ok
/dev/sda2    105906176         1024K   ok
/dev/sdb1        65536           64K   ok
```

Run virt-alignment-scan without any *−a* or *−d* options to scan all libvirt domains.

```
# virt−alignment−scan
F16x64:/dev/sda1      1048576         1024K   ok
F16x64:/dev/sda2      2097152         2048K   ok
F16x64:/dev/sda3    526385152         2048K   ok
```

The output consists of 4 or more whitespace-separated columns. Only the first 4 columns are significant if you want to parse this from a program. The columns are:

col 1

The device and partition name (eg. */dev/sda1* meaning the first partition on the first block device).

When listing all libvirt domains (no *−a* or *−d* option given) this column is prefixed by the libvirt name or UUID (if *−−uuid* is given). eg: `WinXP:/dev/sda1`

col 2

the start of the partition in bytes

col 3
: the alignment in bytes or Kbytes (eg. `512` or `4K`)

col 4
: `ok` if the alignment is best for performance, or `bad` if the alignment can cause performance problems

cols 5+
: optional free-text explanation.

The exit code from the program changes depending on whether poorly aligned partitions were found. See ‘‘EXIT STATUS’’ below.

If you just want the exit code with no output, use the *−q* option.

OPTIONS

−−help
: Display brief help.

−a file
−−add file
: Add *file* which should be a disk image from a virtual machine.

  The format of the disk image is auto-detected. To override this and force a particular format use the *−−format=*.. option.

−a URI
−−add URI
: Add a remote disk. See ‘‘ADDING REMOTE STORAGE’’ in **guestfish** (1).

−−blocksize=512
−−blocksize=4096
−−blocksize
: This parameter sets the sector size of the disk image. It affects all explicitly added subsequent disks after this parameter. Using*−−bloc ksize* with no argument switches the disk sector size to the default value which is usually 512 bytes. See also ‘‘guestfs_add_drive_opts’’ in **guestfs** (3).

−c URI
−−connect URI
: If using libvirt, connect to the given *URI*. If omitted, then we connect to the default libvirt hypervisor.

  If you specify guest block devices directly (*−a*), then libvirt is not used at all.

−d guest
−−domain guest
: Add all the disks from the named libvirt guest. Domain UUIDs can be used instead of names.

−−format=raw|qcow2|..
−−format
: The default for the *−a* option is to auto-detect the format of the disk image. Using this forces the disk format for *−a* options which follow on the command line. Using *−−format* with no argument switches back to auto-detection for subsequent *−a* options.

  For example:

```
virt−alignment−scan −−format=raw −a disk.img
```

  forces raw format (no auto-detection) for *disk.img*.

```
virt−alignment−scan −−format=raw −a disk.img −−format −a another.img
```

  forces raw format (no auto-detection) for *disk.img* and reverts to auto-detection for *another.img*.

  If you have untrusted raw-format guest disk images, you should use this option to specify the disk format. This avoids a possible security problem with malicious guests (CVE−2010−3851).

−P nr_threads
Since libguestfs 1.22, virt-alignment-scan is multithreaded and examines guests in parallel. By default the number of threads to use is chosen based on the amount of free memory available at the time that virt-alignment-scan is started. You can force virt-alignment-scan to use at most `nr_threads` by using the *−P* option.

Note that *−P 0* means to autodetect, and *−P 1* means to use a single thread.

−q
−−quiet
Don't produce any output. Just set the exit code (see ''EXIT STATUS'' below).

−−uuid
Print UUIDs instead of names. This is useful for following a guest even when the guest is migrated or renamed, or when two guests happen to have the same name.

This option only applies when listing all libvirt domains (when no *−a* or *−d* options are specified).

−v
−−verbose
Enable verbose messages for debugging.

−V
−−version
Display version number and exit.

−x Enable tracing of libguestfs API calls.

RECOMMENDED ALIGNMENT

Operating systems older than Windows 2008 and Linux before ca.2010 place the first sector of the first partition at sector 63, with a 512 byte sector size. This happens because of a historical accident. Drives have to report a cylinder / head / sector (CHS) geometry to the BIOS. The geometry is completely meaningless on modern drives, but it happens that the geometry reported always has 63 sectors per track. The operating system therefore places the first partition at the start of the second ''track'', at sector 63.

When the guest OS is virtualized, the host operating system and hypervisor may prefer accesses aligned to one of:

- 512 bytes

  if the host OS uses local storage directly on hard drive partitions, and the hard drive has 512 byte physical sectors.

- 4 Kbytes

  for local storage on new hard drives with 4Kbyte physical sectors; for file-backed storage on filesystems with 4Kbyte block size; or for some types of network-attached storage.

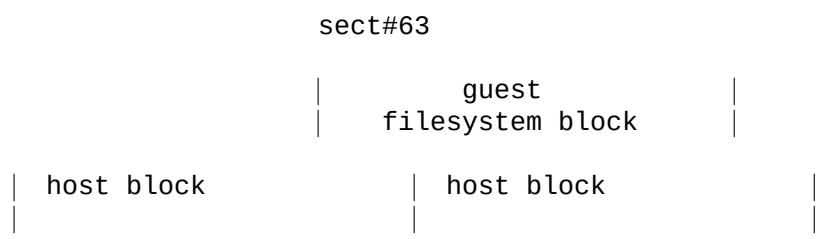
- 64 Kbytes

  for high-end network-attached storage. This is the optimal block size for some NetApp hardware.

- 1 Mbyte

  see ''1 MB PARTITION ALIGNMENT'' below.

Partitions which are not aligned correctly to the underlying storage cause extra I/O. For example:



In this example, each time a 4K guest block is read, two blocks on the host must be accessed (so twice as much I/O is done). When a 4K guest block is written, two host blocks must first be read, the old and new data combined, and the two blocks written back (4x I/O).

LINUX HOST BLOCK AND I/O SIZE

New versions of the Linux kernel expose the physical and logical block size, and minimum and recommended I/O size.

For a typical consumer hard drive with 512 byte sectors:

```
$ cat /sys/block/sda/queue/hw_sector_size
512
$ cat /sys/block/sda/queue/physical_block_size
512
$ cat /sys/block/sda/queue/logical_block_size
512
$ cat /sys/block/sda/queue/minimum_io_size
512
$ cat /sys/block/sda/queue/optimal_io_size
0
```

For a new consumer hard drive with 4Kbyte sectors:

```
$ cat /sys/block/sda/queue/hw_sector_size
4096
$ cat /sys/block/sda/queue/physical_block_size
4096
$ cat /sys/block/sda/queue/logical_block_size
4096
$ cat /sys/block/sda/queue/minimum_io_size
4096
$ cat /sys/block/sda/queue/optimal_io_size
0
```

For a NetApp LUN:

```
$ cat /sys/block/sdc/queue/logical_block_size
512
$ cat /sys/block/sdc/queue/physical_block_size
512
$ cat /sys/block/sdc/queue/minimum_io_size
4096
$ cat /sys/block/sdc/queue/optimal_io_size
65536
```

The NetApp allows 512 byte accesses (but they will be very inefficient), prefers a minimum 4K I/O size, but the optimal I/O size is 64K.

For detailed information about what these numbers mean, see http://docs.redhat.com/docs/en−US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/newstorage−iolim

[Thanks to Matt Booth for providing 4K drive data. Thanks to Mike Snitzer for providing NetApp data and

additional information.]

1 MB PARTITION ALIGNMENT

Microsoft picked 1 MB as the default alignment for all partitions starting with Windows 2008 Server, and Linux has followed this.

Assuming 512 byte sectors in the guest, you will now see the first partition starting at sector 2048, and subsequent partitions (if any) will start at a multiple of 2048 sectors.

1 MB alignment is compatible with all current alignment requirements (4K, 64K) and provides room for future growth in physical block sizes.

SETTING ALIGNMENT

virt−resize (1) can change the alignment of the partitions of some guests. Currently it can fully align all the partitions of all Windows guests, and it will fix the bootloader where necessary. For Linux guests, it can align the second and subsequent partitions, so the majority of OS accesses except at boot will be aligned.

Another way to correct partition alignment problems is to reinstall your guest operating systems. If you install operating systems from templates, ensure these have correct partition alignment too.

For older versions of Windows, the following NetApp document contains useful information: http://media.netapp.com/documents/tr−3747.pdf

For Red Hat Enterprise Linux ≤ 5, use a Kickstart script that contains an explicit `%pre` section that creates aligned partitions using **parted** (8). Do not use the Kickstart`part` command. The NetApp document above contains an example.

EXIT STATUS

This program returns:

- 0

  successful exit, all partitions are aligned ≥ 64K for best performance

- 1

  an error scanning the disk image or guest

- 2

  successful exit, some partitions have alignment < 64K which can result in poor performance on high end network storage

- 3

  successful exit, some partitions have alignment < 4K which can result in poor performance on most hypervisors

SEE ALSO

guestfs (3), **guestfish** (1), **virt−filesystems** (1), **virt−rescue** (1), **virt−resize** (1), http://libguestfs.org/.

AUTHOR

Richard W.M. Jones http://people.redhat.com/~rjones/

COPYRIGHT

LICENSE

to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110−1301 USA.

BUGS

To get a list of bugs against libguestfs, use this link: https://bugzilla.redhat.com/buglist.cgi?component=libguestfs&product=Virtualization+Tools

To report a new bug against libguestfs, use this link: https://bugzilla.redhat.com/enter_bug.cgi?component=libguestfs&product=Virtualization+Tools

When reporting a bug, please supply:

- The version of libguestfs.
- Where you got libguestfs (eg. which Linux distro, compiled from source, etc)
- Describe the bug accurately and give a way to reproduce it.
- Run **libguestfs−test−tool** (1) and paste the **complete, unedited** output into the bug report.