Multiple devices (md, software RAID)

Introduction

md is a Linux kernel facility with user space supporting tools and files.

md was originally "mirror disk". It was renamed "multiple devices" when functionality was extended to more than mirrored disks. Functionality now includes non-mirrored RAID and multipathing. At the time of writing, Blue Light had only used md to create RAID 1 devices.

md assembles multiple devices into a single virtual device.

The multiple devices are variously called "underlying devices" and "RAID devices" in the documentation. On this WIKI page they are always "underlying devices".

md device file names: /dev/md<n> and /dev/md/<n>)

Originally md device files were /dev/md[[:digit:]]+. Now /dev/md**/**[[:digit:]]+ is preferred but does not work on at least Ubuntu 12.04 Precise.

⚠️ Andrey reports that having auto-generated /dev/md/* names in mdadm.conf breaks GRUB or initramfs in recent (Trusty and Jessie) systems. TODO: clarify (including adding related bug references) and determine whether our policy should be to defend against such auto-generated names or whether this is a transient situation which will be bug-passed.

RAID 1

Creation

Create the block devices to be used as underlying devices. Typically these are HDD partitions but any block devices can be used. In the following example, /dev/sdb1 /dev/sdc1 are used.

```
mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
```

In case only one underlying device is available (for example during system build or recovery) the same command can be used with the missing underlying devce name replaced by "missing". For example:

```
mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 missing
```

If the md device provides the file system including /boot or is an LVM PV (physical volume) then /etc/mdadm/mdadm.conf must be re-populated as shown below and the initramfs re-generated by:

```
update-initramfs -u
```

EFI System partition (ESP)

EFI only recognizes FAT16/FAT32, so according to most sources, RAID 1 is not possible. However, that is not true: RAID 1 **is** possible as long as metadata version 0.9 or 1.0 is used. The reason it works is because metadata 0.9 or 1.0 is stored at the **end** of the partition, which doesn't interfere with EFI, whereas metadata 1.2 does because it's stored at the beginning.

To convert the EFI System partition device to md RAID ...

Back up the current ESP files and unmount the file system:
```
    cd /boot && tar -cvzf efi.tar efi
    umount /boot/efi
```

Create a RAID array:
```
    mdadm --create --metadata=1.0 --verbose /dev/md1 --level=1 --raid-devices=2 /dev/sda1 missing
```

ℹ "missing" allows you to specify that a second device isn't present **yet** .

Format the partition (you might need to install **dosfsutils** package first to get the command):
```
  mkfs.vfat /dev/md1
```

Update /etc/fstab's /boot/efi line with the new file system's UUID:
```
  blkid | grep md1
```
... and edit /etc/fstab, inserting the UUID into the /boot/efi line.

Restore the ESP files:
```
  cd /boot && tar -xvzf efi.tar
```

Removal

For example:

```
     mdadm --manage /dev/md2 --fail /dev/sd[ab]1
     mdadm --manage /dev/md2 --remove /dev/sd[ab]1
     mdadm --manage /dev/md2 --stop
     mdadm --zero-superblock /dev/sd[ab]1
```

Replacing an underlying device

Typically this is done when an HDD has failed.

DOS partition table (a.k.a MBR)

In case the replacement HDD is the same size and has the same partitions as a working HDD, the partitions can be created and GRUB installed in a single command. Assuming sda is the working HDD, partitioned with an MBR and sdb the replacement:

```
dd if=/dev/sda of=/dev/sdb bs=512 count=1
```

The procedure continues at "All partition table types" below.

GUID partition table (GPT)

Create the new underlying device by partitioning.

All partition table types

Ensure the array is active. In case it is inactive use mdadm --manage --run to activate it. TODO: move the last sentence to a new section on activation.

Regardless of how the underlying block device was created, it can now be added to the md device:

```
     mdadm --manage --add /dev/md/<md device index> /dev/<whatever>
```

For example:

```
     mdadm --manage --add /dev/md/0 /dev/sdb1
```

Disabling an underlying device

This is intended to:

- prevent damage in case the underlying device recovers and fails again
- stop spurious error messages
- prepare for replacement

For example:

```
# mdadm --manage --fail /dev/md1 /dev/sdb2
mdadm: set /dev/sdb2 faulty in /dev/md1
# mdadm --manage --remove /dev/md1 /dev/sdb2
mdadm: hot removed /dev/sdb2 from /dev/md1
# mdadm --zero-superblock /dev/sdb2
mdadm: Unrecognised md component device - /dev/sdb2
[normally works; the example above was because sdb had failed]
# echo "1" > /sys/block/sdb/device/delete
```

Remove an underlying device (md device not in use)

Removing /dev/md1 as an example ...

Find which block devices the md device is built on

```
     mdadm --detail /dev/md1
```

Stop the device:

```
     mdadm --stop /dev/md1
```

For each block device the device is built on (here /dev/sda2):

```
     mdadm --zero-superblock /dev/sda2
```

Booting from an md RAID 1 device

⚠ When the root file system is on a RAID device, either directly or because it is an LV based on a RAID device for PV, to prevent boot hanging when the RAID is degraded:

- Before Jessie (TODO: Trusty same?): use kenel parameter bootdegraded=true
- Jessie (TODO: Trusty same?): create /etc/initramfs-tools/conf.d/mdadm containing "BOOT_DEGRADED=true" and copy /usr/share/initramfs-tools/scripts/local-top/mdadm to /etc/initramfs-tools/scripts/local-top and patch it as described in https://jira.bluelightav.org/browse/FINANCESRV-368?focusedCommentId=53091 then rebuild the initramfs with
  ```
  update-initramfs -u
  ```

When a RAID 1 md device is used for /boot, GRUB should be installed to both underlying devices, not to the md device. This is done using

```
     dpkg-reconfigure grub-pc
```

GRUB2 can be configured to include md support so can assemble the /boot md before loading initrd (or the kernel directly) but Blue Light accepts what the distro delivers.

Routine maintenance

checkarray

At least Debian 7 Wheezy has /etc/cron.d/mdadm which runs

```
     /usr/share/mdadm/checkarray --cron --all --idle --quiet
```

on the 14th of each month at 02:19.

The command only checks the md arrays, it does not fix them. For each array /dev/md/<n>, it results in messages in syslog like:

```
RebuildStart event detected on md device ...
Rebuild(20|40|60|80) event detected on md device ...
RebuildFinished event detected on md device ...
```

Despite "Rebuild" it is actually a check.

For small arrays the "`Rebuild(20|40|60|80)` ..." progress indicators are not generated. We have seen some of those messages missing.

Sometimes mismatches are reported. According to https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=518834#38, that is a problem for RAID 5 but not for RAID 1. Presumably it is a problem for RAID 6 too.

Re-populating /etc/mdadm/mdadm.conf

/etc/mdadm/mdadm.conf is generated when md arrays are set up changes may require updating it by:

```
/usr/share/mdadm/mkconf > /etc/mdadm/mdadm.conf
```

It is prudent to rebuild the initramfs after changing mdadm.conf:

```
update-initramfs -u
```

mdadm syntax table

In case mdadm --help and the mdam man page are not what is needed.

The man page says the general syntax is

```
  mdadm [mode] [options] <component-devices>
```

But that does not agree with mdadm --help output which lists ...

```
    mdadm mode device options...
    mdadm mode [option ...] devices
```

... and notes that the --manage mode can be omitted.

⚠ BusyBox mdam commands are more consistent than the full commands; options always come before devices.

Mode	device before options	Options	Notes
--assemble	Y	bitmap, uuid, super-minor, name, config, scan, run, force, update, no-degraded	
--build	Y		
--create	Y		
--grow	N		
--incremental	No options		
--misc	N		
--manage	Y	add, remove, fail, set-faulty, run, stop, readonly, readwrite	--manage may be omitted
--monitor	No device		

Troubleshooting

Degraded array report

The system my detect a problem with the array and drop an underlying device. This is normally reported by mail and by Nagios.

It can be confirmed and the excluded underlying device (partition) identified by `cat /proc/mdstat`

Defects in the physical device may be identified in kern.log and by `smartctl -a /dev/sd<letter>`

In case smartctl output shows extended tests are not being run (old build computers), update /etc/smarttd.conf from git and edit to suit.

In case no defects are found, there is no choice but to add the excluded underlying device back into the md device as documented under "Replacing an underlying device" above.

If defects are found, ensure the underlying device is not used again by zeroing its md superblock as documented under "Remove an underlying device (md device not in use)" above.

⚠ In one case (bafi), during boot an array was assembled from a defective underlying device and the good underlying device excluded.

initrd fails to assemble md devices

When initrd is responsible for assembling the md devices, it needs to be rebuilt when the md devices (not their underlying devices) change, for example when an md is removed and a new one created with the same name.

If this has not been done and the md devices were required to boot, a busybox prompt is presented. `mdadm --assemble --scan` was not effective. The md device had to be created by running `mdadm --assemble /dev/md1 /dev/sda2 /dev/sdb2`

When the system has been booted, the damage can be fixed by:

1. Updating /etc/mdadm/mdadm.conf to include current md UUIDs (updated lines can be generated by running `mdadm --detail --scan`).
2. Updating the current initrd by running `update-initramfs -u`

LVM PVs on missing md devices

After fixing the missing md devices as above, the VGs and LVs can be enabled by running the busybox command `lvm` and then using its help to find the commands to do so (cut down versions of the GNU Linux equivalents).

Messages

auto-read-only

Seen in /proc/mdstat. Normal until after the first write to the device (can be forced).

/etc/mdadm/mdadm.conf defines no arrays

Populate /etc/mdadm/mdadm.conf as described above.

nvidia: wrong # of devices in RAID set "nvidia_gbbagfad"

This is normal on some motherboards with an Nvidia chipset with BIOS-provided RAID which, even when disabled, seems to lead to these messages. We have not found a way to disable them.

RebuildFinished event detected on md device *, component device mismatches found: * (on raid level *)

A non-zero number of mismatches is OK on RAID 1.

References

- Overview: md man page
- mdadm man page
- mdadm.conf man page
- kernel.org WIKI: https://raid.wiki.kernel.org
- Wikiversity (nice summary): https://en.wikiversity.org/wiki/Linux/mdadm