B.1. Characteristics of Linux RAID levels

All RAID levels are used to combine multiple devices into a single MD array. The MD plug-in is a region-manager, so EVMS refers to MD arrays as "regions." MD can create these regions using disks, segments or other regions. This means that it's possible to create RAID regions using other RAID regions, and thus combine multiple RAID levels within a single volume stack.

The following subsections describe the characteristics of each Linux RAID level. Within EVMS, these levels can be thought of as sub-modules of the MD plug-in.

B.1.1. Linear mode

Linear-RAID regions combine objects by appending them to each other. Writing (or reading) linearly to the MD region starts by writing to the first child object. When that object is full, writes continue on the second child object, and so on until the final child object is full. Child objects of a Linear-RAID region do not have to be the same size.

Advantage:

- Linear-RAID provides a simple method for building very large regions using several small objects.

Disadvantages:

- Linear-RAID is not "true" RAID, in the sense that there is no data redundancy. If one disk crashes, the RAID region will be unavailable, and will result in a loss of some or all data on that region.
- Linear-RAID provides little or no performance benefit. The objects are combined in a simple, linear fashion that doesn't allow for much (if any) I/O in parallel to multiple child objects. The performance of a Linear-RAID will generally be equivalent to the performance of a single disk.

B.1.2. RAID-0

RAID-0 is usually referred to as "striping." This means that data in a RAID-0 region is evenly distributed and interleaved on all the child objects. For example, when writing 16 KB of data to a RAID-0 region with three child objects and a chunk-size of 4 KB, the data would be written as follows:

- 4 KB to object 0
- 4 KB to object 1
- 4 KB to object 2
- 4 KB to object 0

Advantages:

- Like Linear-RAID, RAID-0 provides a simple method for building very large regions using several small objects.

- In general, RAID-0 provides I/O performance improvements, because it can break large I/O requests up and submit them in parallel across several disks.

Disadvantage:

- Also like Linear-RAID, RAID-0 is not "true" RAID, in the sense that there is no data redundancy (hence the name RAID "zero"). If one disk crashes, the RAID region will be unavailable, and will likely result in a loss of all data on that region.

B.1.3. RAID-1

RAID-1 is usually referred to as "mirroring." Each child object in a RAID-1 region contains an identical copy of the data in the region. A write to a RAID-1 region results in that data being written simultaneously to all child objects. A read from a RAID-1 region can result in reading the data from any one of the child objects. Child objects of a RAID-1 region do not have to be the same size, but the size of the region will be equal to the size of the smallest child object.

Advantages:

- RAID-1 provides complete data redundancy. In a RAID-1 region made from N child objects, up to N-1 of those objects can crash and the region will still be operational, and can retrieve data from the remaining objects.

- RAID-1 can provide improved performance on I/O-reads. Because all child objects contain a full copy of the data, multiple read requests can be load-balanced among all the objects.

Disadvantages:

- RAID-1 can cause a decrease in performance on I/O-writes. Because each child object must have a full copy of the data, each write to the region must be duplicated and sent to each object. A write request cannot be completed until all duplicated writes to the child objects are complete.

- A RAID-1 region with N disks costs N times as much as a single disk, but only provides the storage space of a single disk.

B.1.4. RAID-4/5

RAID-4/5 is often referred to as "striping with parity." Like RAID-0, the data in a RAID-4/5 region is striped, or interleaved, across all the child objects. However, in RAID-4/5, parity information is also calculated and recorded for each stripe of data in order to provide redundancy in case one of the objects is lost. In the event of a disk crash, the data from that disk can be recovered based on the data on the remaining disks and the parity information.

In RAID-4 regions, a single child object is used to store the parity information for each data stripe. However, this can cause an I/O bottleneck on this one object, because the parity information must be updated for each I/O-write to the region.

In RAID-5 regions, the parity is spread evenly across all the child objects in the region, thus eliminating the parity bottleneck in RAID-4. RAID-5 provides four different algorithms for how the parity is distributed. In fact, RAID-4 is often thought of as a special case of RAID-5 with a parity algorithm that simply uses one object instead of all objects. This is the viewpoint that Linux and EVMS use. Therefore, the RAID-4/5 level is often just referred to as RAID-5, with RAID-4 simply being one of the five available parity algorithms.

Advantages and disadvantages

- Like RAID-1, RAID-4/5 provides redundancy in the event of a hardware failure. However, unlike RAID-1, RAID-4/5 can only survive the loss of a single object. This is because only one object's worth of parity is recorded. If more than one object is lost, there isn't enough parity information to recover the lost data.

- RAID-4/5 provides redundancy more cost effectively than RAID-1. A RAID-4/5 region with N disks provides N-1 times the storage space of a single disk. The redundancy comes at the cost of only a single disk in the region.

- Like RAID-0, RAID-4/5 can generally provide an I/O performance improvement, because large I/O requests can be broken up and submitted in parallel to the multiple child objects. However, on I/O-writes the performance improvement will be less than that of RAID-0, because the parity information must be calculated and rewritten each time a write request is serviced. In addition, in order to provide any performance improvement on I/O-writes, an in-memory cache must be maintained for recently accessed stripes so the parity information can be quickly recalculated. If a write request is received for a stripe of data that isn't in the cache, the data chunks for the stripe must first be read from disk in order to calculate the parity. If such cache-misses occur too often, the I/O-write performance could potentially be worse than even a Linear-RAID region.

B.1.5. Multipath

A multipath region consists of one or more objects, just like the other RAID levels. However, in multipath, the child objects actually represent multiple physical paths to the same physical disk. Such setups are often found on systems with fiber-attached storage devices or SANs.

Multipath is not actually part of the RAID standard, but was added to the Linux MD driver because it provides a convenient place to create "virtual" devices that consist of multiple underlying devices.

The previous RAID levels can all be created using a wide variety of storage devices, including generic, locally attached disks (for example, IDE and SCSI). However, Multipath can only be used if the hardware actually contains multiple physical paths to the storage device, and such hardware is usually available on high-end systems with fiber-or network-attached storage. Therefore, if you don't know whether you should be using the Multipath module, chances are you don't need to use it.

Like RAID-1 and RAID-4/5, Multipath provides redundancy against hardware failures. However, unlike these other RAID levels, Multipath protects against failures in the paths to the device, and not failures in the device itself. If one of the paths is lost (for example, a network adapter breaks or a fiber-optic cable is removed), I/O will be redirected to the remaining paths.

Like RAID-0 and RAID-4/5, Multipath can provide I/O performance improvements by load balancing I/O requests across the various paths.

Prev Home Next
The MD region manager Up Creating an MD region