NAME

md - memory disk

SYNOPSIS

To compile this driver into the kernel, place the following lines in your kernel configuration file:

```
device md
```

Alternatively, to load the driver as a module at boot time, place the following line in loader.conf(5):

```
geom_md_load="YES"
```

DESCRIPTION

The **md** driver provides support for four kinds of memory backed virtual disks:

malloc Backing store is allocated using malloc(9). Only one malloc-bucket is used, which means that all **md** devices with **malloc** backing must share the malloc-per-bucket-quota. The exact size of this quota varies, in particular with the amount of RAM in the system. The exact value can be determined with vmstat(8).

preload A module loaded by loader(8) with type ‘md_image’ is used for backing store. For backwards compatibility the type ‘mfs_root’ is also recognized. See the description of module loading directives in loader.conf(5) and note that the module name will either be an absolute path to the image file or the name of a file in the *module_path*.

If the kernel is created with option MD_ROOT the first preloaded image found will become the root file system.

vnode A regular file is used as backing store. This allows for mounting ISO images without the tedious detour over actual physical media.

swap Backing store is allocated from buffer memory. Pages get pushed out to the swap when the system is under memory pressure, otherwise they stay in the operating memory. Using **swap** backing is generally preferable over **malloc** backing.

For more information, please see mdconfig(8).

EXAMPLES

To create a kernel with a ramdisk or MD file system, your kernel config needs the following options:

```
options    MD_ROOT                    # MD is a potential root device
options    MD_ROOT_READONLY    # disallow mounting root writeable
options    MD_ROOT_SIZE=8192       # 8MB ram disk
makeoptions          MFS_IMAGE=/h/foo/ARM-MD
options    ROOTDEVNAME=\"ufs:md0\"
```

The image in */h/foo/ARM-MD* will be loaded as the initial image each boot. To create the image to use, please follow the steps to create a file-backed disk found in the mdconfig(8) man page. Other tools will also create these images, such as NanoBSD.

ARM KERNEL OPTIONS

On armv6 and armv7 architectures, an MD_ROOT image larger than approximately 55 MiB may require building a custom kernel using several tuning options related to kernel memory usage.

options LOCORE_MAP_MB=<num>

This configures how much memory is mapped for the kernel during the early initialization stages. The value must be at least as large as the kernel plus all preloaded modules, including the root image. There is no downside to setting this value too large, as long as it does not exceed the amount of physical memory. The default is 64 MiB.

options NKPT2PG=<num>

This configures the number of kernel L2 page table pages to preallocate during kernel initialization. Each L2 page can map 4 MiB of kernel space. The value must be large enough to map the kernel plus all preloaded modules, including the root image. The default value is 32, which is sufficient to map 128 MiB.

options VM_KMEM_SIZE_SCALE=<num>

This configures the amount of kernel virtual address (KVA) space to dedicate to the kmem_arena map. The scale value is the ratio of physical to virtual pages. The default value of 3 allocates a page of KVA for each 3 pages of physical ram in the system. The kernel and modules, including the root image, also consume KVA. The combination of a large root image and the default scaling may preallocate so much KVA that there is not enough remaining address space to allocate kernel stacks, IO buffers, and other resources that are not part of kmem_arena. Overallocating kmem_arena space is likely to manifest as failure to launch userland processes with "cannot allocate kernel stack" messages. Setting the scale value too high may result in kernel failure to allocate memory because kmem_arena is too small, and the failure may require significant runtime to manifest. Empirically, a value of 5 works well for a 200 MiB root image on a system with 2 GiB of physical ram.

SEE ALSO

gpart(8), loader(8), mdconfig(8), mdmfs(8), newfs(8), vmstat(8)

HISTORY

The **md** driver first appeared in FreeBSD 4.0 as a cleaner replacement for the MFS functionality previously used in PicoBSD and in the FreeBSD installation process.

The **md** driver did a hostile takeover of the vn(4) driver in FreeBSD 5.0.

AUTHORS

The **md** driver was written by Poul-Henning Kamp <*phk@FreeBSD.org*>.