Memory File System (MemFS) 1.0 for HP-UX 11i v3

Administrator's Guide

First Edition

Manufacturing Part Number: 5992-5788

December 2008

Legal Notices

Trademark Notices

UNIX is a registered trademark of The Open Group.

Contents

Contents

About This Document

The *Memory File System (MemFS) 1.0 for HP-UX 11i v3 Administrator's Guide* describes how to install, configure, and tune MemFS on HP-UX 11i v3.

Intended Audience

This document is for system administrators responsible for installing and configuring MemFS on HP-UX 11i v3. Readers are expected to have knowledge of:

- HP-UX System administration concepts
- UNIX operating system concepts
- UNIX File system concepts

Document Organization

The *Memory File System (MemFS) 1.0 for HP-UX 11i v3 Administrator's Guide* is divided into the following chapters:

Table 1 Document Organization

Chapter	Description
Chapter 1 "Introducing HP-UX Memory File System (MemFS) 1.0," on page 7	Describes the architecture of HP-UX Memory File System.
Chapter 2 "Installing, Configuring, Tuning, and Removing HP-UX Memory File System," on page 15	Describes the steps for installing, configuring, tuning, and removing Memory File System.

Typographic Conventions

Table 2 describes the typographic conventions used in this document.

Table 2 Typographic Conventions

Typeface	Usage	Examples
monospace	Computer output, files, directories, software elements such as command options, function names, and parameters	Read tunables from the `/etc/vx/tunefstab` file. See the `ls(1)` manpage for more information.

Table 2 Typographic Conventions (Continued)

Typeface	Usage	Examples
italic	New terms, book titles, emphasis, variables replaced with a name or value	See the *HP-UX Memory File System (MemFS) 1.0 Administrator's Guide* for details.
%	C shell prompt	Not applicable
$	Bourne/Korn shell prompt	Not applicable
#	Superuser prompt (all shells)	Not applicable
\	Continued input on the following line; you do not type this character	`# mount -F vxfs \ /h/filesys`
[ ]	In command synopsis, brackets indicates an optional argument.	`ls [ -a ]`
|	In command synopsis, a vertical bar separates mutually exclusive arguments.	`mount [ suid | nosuid ]`
blue text	An active hypertext link	In PDF and HTML files, click on links to move to the specified location.

Related Documentation

- *MemFS -A Memory-based File System on HP-UX 11i v3* whitepaper.

1 Introducing HP-UX Memory File System (MemFS) 1.0

This chapter introduces you to the HP-UX implementation of Memory File System (MemFS).

The following topics are discussed:

- Overview
- Need for a Memory-Based File System
- Architecture of HP-UX Memory File System
- Application Areas for HP-UX Memory File System
- Dependencies

Overview

A Memory-based File System (MemFS) is a file system that resides in memory. It does not normally write data out to stable storage. A MemFS is created from a mount operation, and ceases to exist when it is un-mounted. The purpose of such a file system is to provide fast access for temporary files that do not need to be kept for an indeterminate time. Because it does not normally have to do I/O to stable storage, the MemFS is able to provide extremely high throughput.

Keeping data in memory comes at a cost. It consumes system physical memory. Even with today's large-memory systems, physical memory comes at a premium. A system or application that runs out of available memory at a critical time can cause irreparable loss to the user. This is the reason why most Virtual Memory (VM) management systems implement a paging policy, wherein less frequently used memory pages are paged out to a swap device. This policy has been extended to MemFS. Under memory pressure, the VM system can deallocate MemFS pages and re-assign them where needed.

Unlike the MemFS on HP-UX 11iv2, the 11iv3 version of MemFS does not require a user process to be associated with each of its instances. MemFS pages will be directly written to system swap device under memory pressure and will be retrieved when needed.

Need for a Memory-Based File System

A memory-based file system is typically used as storage for temporary files. By keeping as much data as possible in memory, it avoids having to perform disk I/O and the associated overhead. Traditional file systems manage two types of data. One is the file content (called data), which an application accesses through the read, write and mmap mechanisms. The other, called metadata is the information related to file attributes and the file system structure.

Most file systems implement a buffering mechanism for data. This implies that a copy of the disk data blocks is maintained in memory (buffer cache or page cache, based on the implementation). This buffer is used to service the reads from a disk, and to implement a "delayed write" mechanism. Using a buffer helps the file systems avoid disk I/O to some extent and improve the file system performance. Structural and attribute changes are buffered to a lesser extent, and will usually cause disk I/O to be done.

A memory-based file system goes a bit further to avoid disk I/O. Metadata is always kept in memory, so the overhead of writing structural changes to the disk is totally avoided. MemFS data blocks are treated mostly at par with other file system blocks and can be swapped out of the cache. There is a difference, though. Disk based file systems try to keep the disk copy up to date with the cache copy. So, "dirty" buffers/pages (those that have been modified) are periodically written back to the disk. This is important since these file systems are expected to preserve data across system shutdown and reboot. A memory-based file system has no such requirement, so data is written to the disk only if that buffer/page is being paged out.

Before memory-based file systems became popular, the concept of a "RAM disk" existed. A RAM disk reserves a range of memory and makes it available through a block device interface using a pseudo driver. The block device interface permits a file system to be created on it, in effect providing a memory-based file system. However, RAM disks have certain drawbacks. Since memory is reserved at the time of the disk creation, it is locked from shared system use, whether actually in use or not. Most RAM disks do not support paging of their memory, resulting in very poor system response in cases where the system is running low on free physical memory. Since the system sees a RAM disk as a device rather than a file system, any access to a file stored on it results in a second copy of the data being kept in the file system buffer. Some of the implementations of a RAM disk may provide an improved interface and some of these shortcomings may not be present.

As attractive as the performance aspects sound, a memory-based file system only works efficiently under certain circumstances. The biggest disadvantage is that it does not preserve data across mounts. It cannot be used to replace file systems that store persistent data. Since it uses memory which is a shared resource, excessive use can adversely affect other memory consumers. A typical use of a memory file system is for storing temporary files that are

created, accessed frequently in a short span of time and then deleted. The advantage comes from not having to update large amounts of metadata on file creation - growing the file - and then deletion and cleanup. The write-to-swap feature of memory-based file systems efficiency has direct correlation with the swap activity. As the swap activity increases, the file system performance decreases.

Architecture of HP-UX Memory File System

The memory file-based system which is available as a configurable product in HP-UX 11i v3 is more advanced when compared to the MemFS on HP-UX 11iv2. A file system has the concept of disk layout, which refers to how information such as the superblock, inodes and the data blocks are organized on the disk. HP-UX 11iv3 MemFS has its own new layout, which stores file data in the UFC (Unified File Cache) and metadata in the kernel memory. It does not use the user process address space as backing store for the files data, during memory pressure. It has all the features needed for a basic file system. Some of the more advanced file system features (e.g. journaling, un-buffered I/O) that some of the contemporary file systems provide are neither required nor relevant in the context of a memory file system.

A MemFS is created from a mount (1M) command. The mount operation just creates the necessary data structures which are part of the new layout. It neither pre-allocates the inodes or any memory as the backing store for the file data. The following data structures are created as part of the mount operation:

- Directory entry hash table
- Inode hash table
- The root directory of the mount instance
- VFS Structure

Subsequent to a mount, the MemFS can be used just like any other file system, with the same commands and system calls, as used on other file systems. From an application or user perspective there should be nothing that differentiates it from any other file system, except a better response time.

Internally, MemFS behaves differently from other file systems. Its directory entries are organized in the Directory entry hash table, for faster lookup. The syncer (1M)1 daemon, which periodically flushes "dirty" file system pages to disk, does not have an effect on MemFS pages, as MemFS pages are never added to the syncer dirty list - which is processed periodically for flushing. The metadata of each MemFS instance is allocated from the kernel memory which can not be swapped. MemFS files data is store in UFC. The MemFS pages in

UFC are not treated specially from the pages of other file systems. Periodically, the VM paging algorithm reallocates the less used pages including the MemFS pages from the UFC. When VM paging algorithm decides to reallocate MemFS pages during memory pressure, the MemFS files data will be written directly to system swap device. When ever the files data is needed, the data will be brought back to UFC directly from the system swap. Though HP-UX 11iv3 does not support pageable kernel memory, MemFS and UFC interact with each other to accomplish the above tasks. Thus the HP-UX 11iv3 MemFS does not require the instance specific user process address space for swapping the files data. The overview of MemFS architecture is depicted in the figure shown below (see figure1).

A umount (1M) on the MemFS removes the file system and all its data. As per the specifications, data cannot be recovered once a umount (1M) happens.

MemFS uses two main resources: kernel memory and system swap space. These resources are shared among all the kernel subsystems and user applications. Thus there is a need to wisely use these shared resources. MemFS provides the following tunables to limit the usage exclusively by MemFS:

- memfs_metamax

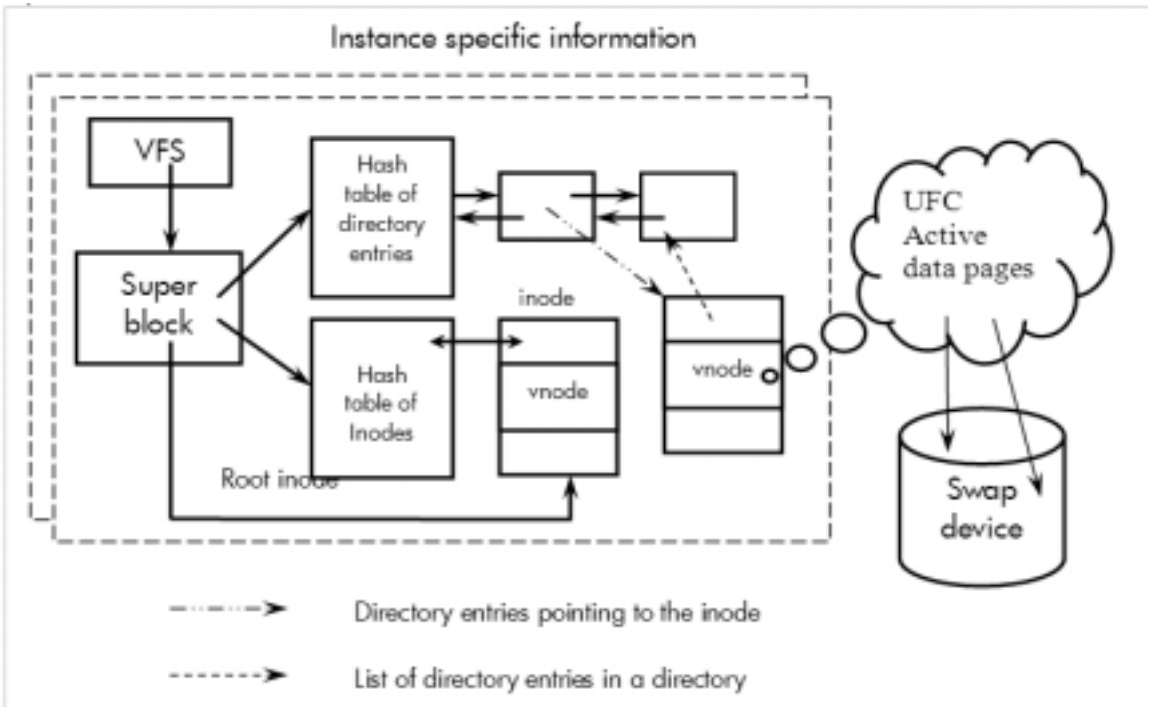
  Tunable to limit the amount of kernel memory used by MemFS

- memfs_swapmax

  Tunable to limit the amount of swap space used by MemFS

For more details on these tunables, see Chapter 2, “Installing, Configuring, Tuning, and Removing HP-UX Memory File System,” on page 15.

Figure 1-1 MemFS Architecture



Application Areas for HP-UX Memory File System

Applications that will benefit most from HP-UX MemFS are those that perform mostly metadata intensive operations like creating and deleting small files and directories. For example compilers, editors and sorting applications. MemFS never writes metadata of files to disk, so directory and file manipulation is always a performance gain. Applications can also interact with MemFS files using the mmap interface.

Dependencies

Following is a list of patches that are required for MemFS:

- PHCO_38751
- PHCO_38752
- PHCO_39038
- PHCO_39039
- PHKL_38077
- PHKL_38949

These patches are available at the following location: http://itrc.hp.com

Without these patches, MemFS cannot be installed. MemFS product should not be installed with `-x enforce_dependencies=false` option of *swinstall(1m)*.

2 Installing, Configuring, Tuning, and Removing HP-UX Memory File System

This chapter describes how to install, configure, tune and remove HP-UX Memory File System (MemFS).

The following topics are discussed:

- Installing HP-UX Memory File System
- Configuring Memory File System
- Tuning Memory File System
- Removing Memory File System

Installing HP-UX Memory File System

You can install HP-UX Memory File System (MemFS) either non-interactively or interactively.

Installing HP-UX MemFS in the Non-Interactive Mode

To install MemFS in non-interactive mode, enter the following command:

```
# swinstall -x reinstall=false -x autoreboot=true -s <depot-path> MemFS
```

Installing MemFS in the Interactive Mode

To install MemFS in interactive mode, enter the following command:

```
# swinstall -x reinstall=false -x autoreboot=true -s <depot-path>
```

Configuring Memory File System

A MemFS instance can be created either through a *mount(1m)* command, or automatically through an entry in the `/etc/fstab` file or by calling *mount(2m)* system call.

Following is the syntax for mounting an MemFS instance using the `mount(1)` command:

```
# /usr/sbin/mount -F memfs [-eQV] [-o size=<size>, ninode=<ninode>, remount]\
directory
```

For example, to create an MemFS instance of size 100MB with 10000 inodes and mounting it on `/memfs`:

```
# /usr/sbin/mount -F memfs -o size=100MB,ninode=10000 /memfs
```

For example, to automatically mount after system reboot, a MemFS instance of size 50MB with 1000 inodes on a mount point, `/memfs` add the following entry to `/etc/fstab` file:

```
memfs /memfs  memfs  size=50MB, ninode=1000 0 0
```

The optional size argument can be used to specify the size of a MemFS instance. Append to size, kb or KB to indicate the value is in kilobytes, mb or MB to indicate megabytes, or gb or GB to indicate gigabytes. If size is not specified or is specified as 0, the size of the file system is not limited and can grow based on the available swap space for MemFS. After mounting a MemFS file system, the mounted instance can be listed using the *bdf(1)/df(1)* command. For example, the above mounted instance will be listed with bdf command, as shown below:

```
Filesystem     kbytes    used   avail    %used   Mounted on
memfs          102400       0    102400    0%     /memfs
```

The swap space is shared by all the MemFS instances in a system and other process running on the system will also consume certain amount of swap space. Because of this, the size argument does not guarantee that the specified size is available for use by a MemFS instance always. If the size option is specified, it just ensures that the file system does not grow beyond the specified value. There can be situations where % used will be much less than 100% and no space left on the swap device to write data. In such situations MemFS displays available space as 0. If size option is not specified or specified as zero, the size of a mounted instance will be the summation of the available free swap for MemFS and the amount of swap space consumed by that particular instance. Depending on the swap usage, the size of a mounted MemFS instance can be varying over time. However, if size option is specified the size of the mounted instance is fixed to the specified value and does not change over time.

The optional ninode argument can be used to specify the maximum number of files that can be created with in a particular instance. If ninode is 0 or the option is not specified the number of files is not restricted, though it is limited by the available memory for MemFS.

The size and ninode values can be modified by remounting the instance with the remount option. For example, to modify the size of the MemFS file system to 100M mounted on /tmp.

```
# /usr/sbin/mount –F memfs -o remount,size=100M /tmp
```

An MemFS instance can be destroyed using the umount(1m) command. Only the directory mount point is required as an argument:

```
# /usr/sbin/umount /memfs
```

Tuning Memory File System

MemFS uses kernel memory for storing the file's metadata. Kernel memory is shared by all the subsystems in the kernel and this memory is not page-able. Over-use of kernel memory can bring down the performance of the system as well as the applications that compete for kernel memory.

MemFS provides a tunable called `memfs_metamax` to restrict the amount of kernel memory used by all MemFS instances.

Writing data to MemFS files result in reserving system swap space for the MemFS files. The system swap space is a shared resource. In addition to MemFS, other applications also consume certain amount of swap space. For example, the execution of very large programs can be affected if MemFS file systems are huge in size. Thus the swap space needs to be used judiciously to allow other applications to have an adequate share of it. MemFS provides a tunable called `memfs_swapmax_pct` to restrict the amount swap space being used by all MemFS instances.

memfs_metamax

The metadata of all the MemFS file systems mounted in the system is stored in kernel memory. The memfs_metamax tunable defines the maximum percentage of kernel memory that can be used to accommodate MemFS metadata. The tunable does not reserve or assure the memory of specified size; it just assures that the metadata does not grow beyond the specified size. Metadata of MemFS file systems is never swapped out of the memory or reused, unless the MemFS file system is unmounted. The value of this tunable can be as low as 1% of kernel memory and as high as 30% kernel memory. The default value is 15% of kernel memory. This tunable can accept both percentage and absolute values.

Values can be specified as follows:

1. Default

   Example:

   `kctune -s memfs_metamax=Default`

2. A percentage of total kernel memory: A positive whole number followed by a percent symbol (for example, 20%).

   Example:

   `kctune -s memfs_metamax=20%`

3. A constant value: A positive whole number that represents number of bytes of kernel memory, optionally followed by a multiplier suffix, where K=1000, KB=1024, M=1000000, MB=(1024*1024), and GB=(1024*1024*1024). For example, a constant value could be 500K.

   Example:

   kctune -s memfs_metamax=100MB

If MemFS utilizes all the memory specified by this tunable, mounting of new MemFS file systems and creation of new files on them will fail and following message will be displayed on console:

```
memfs: metadata exceeds memfs_metamax
```

This tunable can be modified dynamically. If the new limit specified is less than the memory in use by MemFS metadata (for example 20480 bytes), following warning will be displayed on console:

```
Warning: The specified value of tunable memfs_metamax is less than the amount
of memory currently in use by MemFS, 20480 bytes.
```

memfs_swapmax_pct

The `memfs_swapmax_pct` tunable defines the maximum amount of system swap space that can be used by all MemFS instances in a system as percentage of total system swap configured in a system. The tunable does not reserve or assure the swap space of specified size; it just assures that the swap space usage by MemFS does not grow beyond the specified size. The value of this tunable can be as low as 0% of system swap or as high as 80% of system swap. The default value is 50% of system swap.

Tunable values can be specified as follows:

1. Default

   Example:

   kctune -s memfs_swapmax_pct=Default

2. A percentage of total swap space; a positive whole number (0-80)

   Example:

   kctune -s memfs_swapmax_pct=60

Data in MemFS files share the system swap device used by others. Therefore, it is important to define a value for this tunable that balances MemFS swap usage and usage for the rest of the system. If the value is too low, then applications can not issue those writes which results in new block allocations to write data to MemFS files. Tuning this tunable too high and if

MemFS file systems have huge data, can reduce the amount of swap space available to other applications and will negatively impact system performance or applications might fail because of low swap condition.

The value of `memfs_swapmax_pct` may need to be raised on systems with a large amount of data used for MemFS file systems. Writing of data to a file in MemFS file system fails with error `[ENOSPC]` when the swap usage has reached the limit; in addition, the following message will be displayed on the console:

```
memfs: Cumulative data exceeds memfs_swapmax_pct
```

This tunable can be modified dynamically. If the new limit specified is less than the swap space in use by MemFS (for example 10%), following warning will be displayed on console:

```
Warning: The specified value of tunable memfs_swapmax_pct is less than the
percentage of swap space currently in use by MemFS, 10 percent.
```

NOTE The `memfs_metamax` and `memfs_swapmax_pct` tunables are dynamic on HP-UX 11i v3 and can be modified without a system reboot.

Tuning `memfs_metamax` and `memfs_swpmax_pct` to a value less than the current usage is not immediate always, though tunables are set with the new value, the current usage may be more than the newly specified value. However, any operations that try to allocate resources (kernel memory and system swap) will fail if the request goes beyond the limit imposed by current value of the tunables.

Tools Support to Report MemFS Statistics

Tools `sar(1m)` and `GlancePlus` have been enhanced to provide the MemFS statistics such as cumulative memory and swap space consumed by all the instances of MemFS. These tools report `memory/swap` consumed in terms of 4096 bytes (4KB) only, even if `base_pagesize(5)` tunable is tuned to higher values.

sar(1m)

SAR-MEMFS product reports MemFS statistics using `sar(1M)` command. The SAR-MEMFS product provides a new option, `-z` that enables the `sar(1M)` command to report MemFS data.

The `-z` option supports the following parameters:

- `blkcnt`

  Specifies the number of MemFS blocks in the memory.

- `swpcnt`

  Specifies the number of MemFS blocks in the swap.

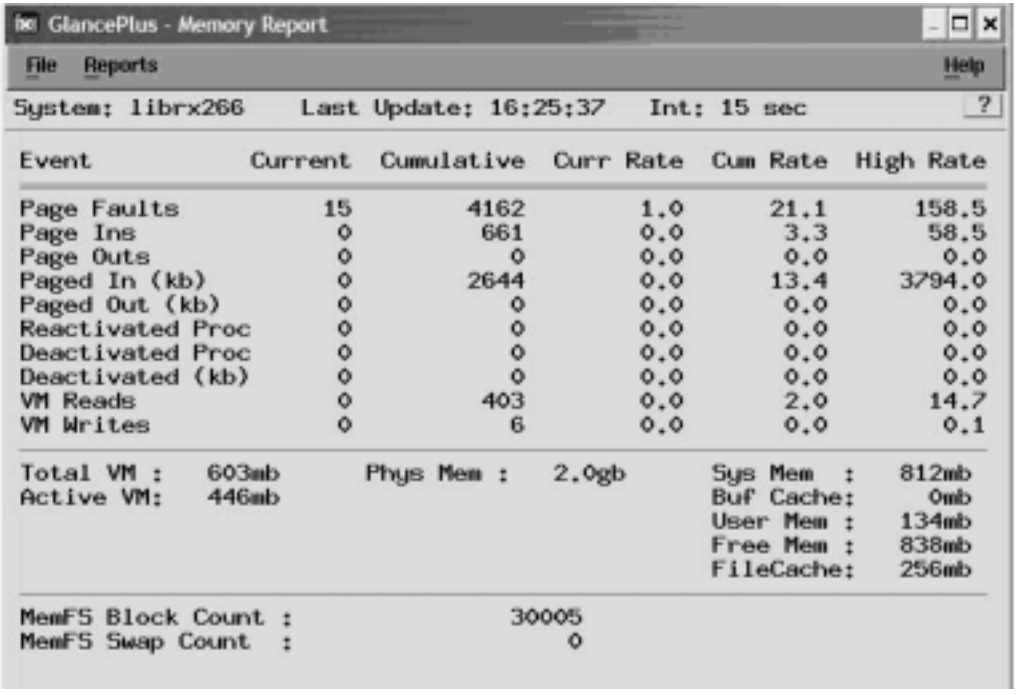
For example:

```
# sar -z 1 10
HP-UX machine B.11.31 U ia64    11/26/08
15:39:59     blkcnt   swpcnt
15:40:00     30031      0
15:40:01     30031      0
15:40:02     30031      0
```

GlancePlus

GlancePlus, C.04.70.000 provides MemFS details similar to `sar(1M)` command. The Memory Report page shows the number of 4KB blocks of MemFS in memory and swap.

Figure 2-1 Memory Report



Removing Memory File System

To remove MemFS, enter the following command:

```
# swremove -x autoreboot=true MemFS
```

Known Problems and Other Limitations

Known Problems

- On large memory usage of MemFS file system instances, such as when MemFS pages occupy memory equal to the total RAM, the system may experience slow responses.

Other Limitations

- Since many HP-UX applications store critical data in `/tmp`, not being able to recover `/tmp` can cause serious damage to the consistency of such applications. HP recommends that you not use MemFS to mount `/tmp` and any file systems which contain critical data that require consistency.
- A limitation with MemFS and Ignite-UX (and other products) is that there is no support for `/tmp` or for `/var/tmp` as MemFS file systems. MemFS file systems should be used only to hold application transient files. The file system and all its contents will be lost with an unmount or reboot operation. Ignite-UX uses the information stored in `/tmp` during the install process and therefore the `/tmp` file systems cannot be MemFS based.