Oracle Linux 9

Configuring Huge Pages

G12279-01
October 2024

Oracle Linux 9 Configuring Huge Pages,

G12279-01

Contents

Preface

Documentation License iv
Conventions iv
Documentation Accessibility iv
Access to Oracle Support for Accessibility iv
Diversity and Inclusion iv

1 About Huge Pages

HugeTLB Pages 1-1
Transparent HugePages 1-1

2 Configuring HugeTLB Pages

Kernel Boot Parameters for HugeTLB Pages 2-1
File-Based Configuration Parameters for HugeTLB Pages 2-2
Requesting HugeTLB Pages by Using Kernel Parameters at Boot Time 2-9
Requesting HugeTLB Pages Using NUMA Node Specific Parameters Early in the Boot Process 2-9
Configuring HugeTLB Pages for a Specific NUMA Node at Runtime 2-11

3 Configuring Transparent HugePages

Parameters Used to Configure Transparent HugePages 3-1
Retrieving the Current Status of Transparent HugePages 3-2
Changing the Current Status of Transparent HugePages 3-3
Changing the defrag Setting of Transparent HugePages 3-4

Preface

Oracle Linux 9: Configuring Huge Pages provides information about configuring Oracle Linux 9 systems to use the kernel huge page feature to increase the page size to improve the performance of applications that have large memory requirements.

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also

mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

About Huge Pages

In Oracle Linux, physical memory is managed in fixed-size blocks called pages. In the x86_64 architecture, the default size of each page is 4 KB.

The kernel stores virtual to physical address mappings for the pages in a data structure known as the page table. However, page table lookups are resource-intensive, so the most recently used addresses are cached in the CPU's Translation Lookaside Buffer (TLB) for faster retrieval. When the CPU needs to fulfill a request for an address-mapping, the CPU first searches the TLB cache. A TLB hit describes the CPU finding the address in the TLB cache. A TLB miss describes the CPU unable to find the requested address-mapping in the cache, in which case the system would perform a resource intensive lookup on the page table to retrieve the address information.

The default page size of 4 KB is suitable for most applications. However, for applications that work with large amounts of memory, the number of 4 KB pages required can be large and can lead to a high number of TLB misses and a performance overhead. Oracle Linux provides huge page features so that applications requiring more memory can have their requirements fulfilled with fewer pages.

HugeTLB Pages

HugeTLB pages are also called static huge pages.

With HugeTLB pages feature, you can reserve pools of huge pages, each of a specified quantity, for each huge page size. The available huge page size options on x86_64 platforms are 2 MB and 1 GB.

For more information about configuring HugeTLB pages, see Configuring HugeTLB Pages

> **Note:**
>
> Best Practices:
>
> - Make requests to the kernel for static huge pages as close to boot time as possible, when the occurrence of memory fragmentation is at a minimum.
> - Huge pages can reduce the amount of memory available to the system. Therefore, when requesting a reserved huge page pool, ensure the pool isn't oversized and that the system's access to memory isn't impacted.

Transparent HugePages

The Transparent HugePages (THP) feature is enabled by default in Oracle Linux. With THP, the kernel automatically assigns huge pages to processes. With THP, you can assign only 2 MB pages on x86_64 platforms.

THP can run in the following modes:

- `system-wide` (default): The kernel assigns huge pages to processes that use large contiguous virtual memory areas whenever it's possible to do so.
- `per-process`: The kernel only assigns huge pages to application processes that explicitly request huge pages through the `madvise()` system call.

For more information about configuring THP, see Configuring Transparent HugePages

2

Configuring HugeTLB Pages

You can configure HugeTLB pages by using the following types of parameters:

- Kernel boot parameters
- File-based configuration parameters

The following sections discuss the parameters in greater detail.

- Kernel Boot Parameters for HugeTLB Pages
- File-Based Configuration Parameters for HugeTLB Pages
- Requesting HugeTLB Pages by Using Kernel Parameters at Boot Time
- Requesting HugeTLB Pages Using NUMA Node Specific Parameters Early in the Boot Process
- Configuring HugeTLB Pages for a Specific NUMA Node at Runtime

Kernel Boot Parameters for HugeTLB Pages

The kernel boot options enable you to specify values such as the size and the number of pages to be reserved in the kernel's pool. Using the kernel boot parameters is the most reliable method of requesting huge pages.

The following table describes the kernel boot parameters available for HugeTLB page setup.

Table 2-1 The Kernel Boot Command Line Parameters for Requesting HugeTLB Pages

Parameters	Purpose	Accepted Value Option on x86_64 Architecture
`default_hugepagesz`	Defines the default size of persistent huge pages configured in the kernel at boot time.	`2M`(default), `1G`

Table 2-1 (Cont.) The Kernel Boot Command Line Parameters for Requesting HugeTLB Pages

Parameters	Purpose	Accepted Value Option on x86_64 Architecture

`hugepagesz` and `hugepages`	Size parameter `hugepagesz` is used with quantity parameter `hugepages` to reserve a pool of a specified page size and quantity. For example, to request a pool of 1500 pages of size 2 MB, the command line options would be as follows:
	`hugepagesz=2M`
	`hugepages=1500`
	If multiple huge page sizes are supported, the "`hugepagesz=`*`<size>`*
	`hugepages=`*`<qty>`*" pair can be specified multiple times, once for each page size. For example, you can use the following command line options to request one pool of four pages of 1 GB size, and a second pool of 1500 pages of 2 MB size:
	`hugepagesz=1G`
	`hugepages=4`
	`hugepagesz=2M`
	`hugepages=1500`
		`Hugepagesz`: `2M`, `1G`
		`hugepages`: `0` or greater

> **Note:**
>
> In a NUMA system, pages reserved with kernel command line options, as shown in the previous table, are divided equally between the NUMA nodes.
> If the requirement is to have a different number of pages on each node, you can use the file-based HugeTLB parameters in the `sysfs` file system. See File-Based Configuration Parameters for HugeTLB Pages and Requesting HugeTLB Pages Using NUMA Node Specific Parameters Early in the Boot Process.

File-Based Configuration Parameters for HugeTLB Pages

The file-based configuration parameters provide runtime access to the configuration settings.

> **Note:**
>
> In addition to accessing the settings at runtime, you can also initialize the parameters early in the boot process, for example, by creating a start-up bash script or by setting the parameters up in a local rc init script.

Multiple instances of each file-based parameter can be configured on a system. For example, on a system that can handle both 2 MB and 1 GB HugeTLB page sizes, several `nr_hugepages` settings can exist. This parameter defines the number of pages in a pool, including the following:

- File `/sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages` for the number of pages in the pool of 2 MB pages.
- File `/sys/kernel/mm/hugepages/hugepages-1048576kB/nr_hugepages` for the number of pages in the pool of 1 GB pages.

The following table outlines commonly used HugeTLB configuration parameters and the multiple file instances that you might find for each parameter.

Table 2-2 Commonly Used File-Based HugeTLB Parameters

Parameter	Purpose	File Paths for Different Instances
`nr_hugepages`	• Each instance of `nr_hugepages` defines the current number of huge pages in the pool associated with that instance. • Can be changed at runtime. • Example command: `echo 20 | sudo tee /proc/sys/vm/nr_hugepages` • Default value is `0`.	The file path formats for different instances of `nr_hugepages` are as follows: • File location: `/proc/sys/vm/nr_hugepages` (present on all systems). • File location: `/sys/kernel/mm/hugepages/hugepages-<SIZE>kB/nr_hugepages` (present on systems that support more than one huge page size). • File location: `/sys/devices/system/node/node{0,1,2…n}/hugepages/hugepages-<SIZE>kB/nr_hugepages` (present on NUMA systems only). **Note:** Use the NUMA node specific path format if you need to request different quantities of pages of different sizes to be supported on specific NUMA nodes. If you use any other path format (for example, `/proc/sys/vm/nr_hugepages`) to request HugeTLB pages. The pages are divided equally between the NUMA nodes.

Table 2-2 (Cont.) Commonly Used File-Based HugeTLB Parameters

Parameter	Purpose	File Paths for Different Instances
`nr_overcommit_hugepages`	• Each instance of `nr_overcommit_hugepages` defines the *additional* number of huge pages that's higher then the quantity specified by `nr_hugepages`. It can be created by the system at runtime through overcommitting memory. • As these additional huge pages become unused, they're freed and returned to the kernel's normal page pool. • Example command: `echo 20 | sudo tee /proc/sys/vm/nr_overcommit_hugepages`	The file path formats for different instances of `nr_overcommit_hugepages` are as follows: • File location `/proc/sys/vm/nr_overcommit_hugepages` (present on all systems). • File location: `/sys/kernel/mm/hugepages/hugepages-<SIZE>kB/nr_overcommit_hugepages` (present on systems that support more than one huge page size). **Note:** The `nr_overcommit_hugepages` parameter isn't defined at the individual node level, so no nod

Table 2-2 (Cont.) Commonly Used File-Based HugeTLB Parameters

Parameter	Purpose	File Paths for Different Instances
		e spe cifi c file exi sts for this sett ing.
`free_hugepages`	• Read-only parameter. • Each instance of `free_hugepages` returns the number of huge pages in its associated page pool that have yet to be allocated.	The file path formats for different instances of `free_hugepages` are as follows: • File location: `/sys/kernel/mm/hugepages/hugepages-<SIZE>kB/free_hugepages` (present on systems that support more than one huge page size). • File location: `/sys/devices/system/node/node{0,1,2…n}/hugepages/hugepages-<SIZE>kB/free_hugepages` (present on NUMA systems only).

Table 2-2 (Cont.) Commonly Used File-Based HugeTLB Parameters

Parameter	Purpose	File Paths for Different Instances
`surplus_hugepages`	• Read-only parameter. • Each instance of `surplus_hugepages` returns the number of huge pages that have been overcommitted from its associated page pool.	The file path formats for different instances of `surplus_hugepages` are as follows: • File location: `/sys/kernel/mm/hugepages/`**`hugepages-<SIZE>kB`**`/surplus_hugepages` (present on systems that support more than one huge page size). • File location: `/sys/devices/system/node/`**`node{0,1,2…n}`**`/hugepages/`**`hugepages-<SIZE>kB`**`/surplus_hugepages` (present on NUMA systems only).

The following sections show file branches under which different instances of the HugeTLB parameters are stored:

/proc/sys/vm

All systems that support static huge pages contain HugeTLB parameter files under `/proc/sys/vm`.

> **Note:**
>
> On many systems, including many Oracle database servers, the `procfs` file system is the main parameter-set used.
>
> The `sysctl` parameter `vm.nr_hugepages` that's commonly initialized in scripts that request huge pages also writes to the `procfs` file `/proc/sys/vm/nr_hugepages`.

The following are example folders under branch `/proc/sys/vm`:

```
├── ...
├── ...
├── nr_hugepages
├── ...
├── nr_overcommit_hugepages
├── ...
├── ...
```

/sys/kernel/mm/hugepages/

Systems that support multiple size pools contain HugeTLB parameter files in size-specific folders under `/sys/kernel/mm/hugepages/`.

The following are example folders under branch `/sys/kernel/mm/hugepages/`:

```
└── hugepages-2048kB
    ├── free_hugepages
    ├── nr_hugepages
    ├── ...
    ├── nr_overcommit_hugepages
    ├── ...
    └── surplus_hugepages

└── hugepages-1048576kB
    ├── free_hugepages
    ├── nr_hugepages
    ├── ...
    ├── nr_overcommit_hugepages
    ├── ...
    └── surplus_hugepages
```

/sys/devices/system/node/

Only NUMA systems contain HugeTLB parameter files under /`sys/devices/system/node/`.

The following are example folders under branch `/sys/devices/system/node`:

```
├  ...
├── node0
│   ├── ...
│   ├──hugepages
│           hugepages-2048kB
│               ├── free_hugepages
│               ├── nr_hugepages
│               └── surplus_hugepages
│
│           hugepages-1048576kB
│               ├── free_hugepages
│               ├── nr_hugepages
│               └── surplus_hugepages
├── node1
    ├── ...
    ├──hugepages
           hugepages-2048kB
               ├── free_hugepages
               ├── nr_hugepages
               └── surplus_hugepages

           hugepages-1048576kB
               ├── free_hugepages
               ├── nr_hugepages
               └── surplus_hugepages
```

Requesting HugeTLB Pages by Using Kernel Parameters at Boot Time

The precise way to request huge pages at boot time depends upon the system's requirements. The following procedure provides some guidance but isn't exclusive of other approaches to configuring boot options.

The following procedure shows how to set default kernel command line options in your GRUB 2 configuration to specify two pools of HugeTLB pages and a default page size on a system that handles multiple huge page sizes. In this procedure, the following are requested:

- A default page size of 1 GB.
- One pool with four HugeTLB pages of 1 GB size.
- One pool of 1500 HugeTLB pages of 2 MB size.

Before beginning the following procedure, ensure that you have the administrative privileges required.

For more information about configuring kernel command line options and GRUB 2 see Oracle Linux 9: Managing Kernels and System Boot

1. Edit the default kernel command line options in the system's GRUB 2 configuration.

   Specify 1 GB size for kernel boot parameter `default_hugepagesz` and 2 pairs of "`hugepagesz=<Size_num>G hugepages=Qty_num`" parameters for the two huge page pools.

   Append the following line to the kernel command line options in `/etc/default/grub`

   ```
   default_hugepagesz=1G hugepagesz=1G hugepages=4 hugepagesz=2M
   hugepages=1500
   ```

2. Regenerate the `GRUB2` configuration file.

   - If the system uses BIOS firmware, run the following command:

     ```
     sudo grub2-mkconfig -o /boot/grub2/grub.cfg
     ```

   - If the system uses UEFI framework, run the following command:

     ```
     sudo grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
     ```

   The next time the system boots, the two huge page pools are requested.

Requesting HugeTLB Pages Using NUMA Node Specific Parameters Early in the Boot Process

The precise way to request huge pages at boot time depends upon the system's requirements. The following procedure provides some guidance but isn't exclusive of other approaches to configuring boot options.

Huge Pages requested by using the kernel boot-time parameters, as shown in the previous example, are divided equally between the NUMA nodes.

However, you might need to request a different number of huge pages for specific nodes by setting the configuration values in a node specific file path. The file path is defined as follows:.

```
/sys/devices/system/node/node{0,1,2…n}/hugepages/hugepages-<SIZE>kB/
```

The following procedure describes how to reserve 299 pages of 2 MB size on node 0, and 300 pages of 2 MB size on node 1 on a NUMA system. This approach uses a custom systemd service to run a shell script after boot, to set the `sysfs` parameters required.

Before beginning the following procedure, ensure that you have the administrative privileges required for all the steps.

For more information about using systemd units, see see Oracle Linux 9: Managing the System With systemd.

1. Create a script file called `hugetlb-reserve-pages.sh` in the `/usr/lib/systemd/` directory and add the following content.

```
#!/bin/sh

nodes_path=/sys/devices/system/node/
if [ ! -d $nodes_path ]; then
    echo "ERROR: $nodes_path does not exist"
    exit 1
fi

########################################################
#                                                      #
#     FUNCTION                                         #
#           reserve_pages <number_of_pages> <node_id>  #
#                                                      #
########################################################

reserve_pages()
{
    echo $1 > $nodes_path/$2/hugepages/hugepages-2048kB/nr_hugepages
}

reserve_pages 299 node0
reserve_pages 300 node1
```

2. Make the script executable:

```
sudo chmod +x /usr/lib/systemd/hugetlb-reserve-pages.sh
```

3. Create a service file called `hugetlb-gigantic-pages.service` in the `/usr/lib/systemd/system/` directory and add the following content to it.

```
[Unit]
Description=HugeTLB Gigantic Pages Reservation
DefaultDependencies=no
Before=dev-hugepages.mount
ConditionPathExists=/sys/devices/system/node

[Service]
Type=oneshot
```

```
RemainAfterExit=yes
ExecStart=/usr/lib/systemd/hugetlb-reserve-pages.sh

[Install]
WantedBy=sysinit.target
```

4. Enable the service file.

```
sudo systemctl enable hugetlb-gigantic-pages
```

Configuring HugeTLB Pages for a Specific NUMA Node at Runtime

In certain cases, you might need make a request for huge pages at runtime.

The following procedure shows how to request 20 HugeTLB pages of size 2048 kB for node2 at runtime.

Before starting, you must ensure you have the required administrative privileges required for all the steps.

1. Run the `numastat` command to show memory statistics relating to the NUMA nodes:

```
numastat -cm | egrep 'Node|Huge'| grep -v AnonHugePages


                 Node 0 Node 1 Node 2 Node 3  Total add
HugePages_Total       0      0      0      0      0
HugePages_Free        0      0      0      0      0
HugePages_Surp        0      0      0      0      0
```

2. Add the required number of huge pages of a specified size to the selected node, for example 20 pages of 2 MB size on node 2:

```
echo 20 | sudo tee /sys/devices/system/node/node2/hugepages/
hugepages-2048kB/nr_hugepages
```

3. Run the `numastat` command again to ensure the request was successful and that the requested memory (in our example 20 x 2 MB pages = 40 MB) has been added `HugePages_Total` for `node2`:

```
numastat -cm | egrep 'Node|Huge'| grep -v AnonHugePages


                 Node 0 Node 1 Node 2 Node 3  Total
HugePages_Total       0      0     40      0     40
HugePages_Free        0      0     40      0     40
HugePages_Surp        0      0      0      0      0
```

3
Configuring Transparent HugePages

The Transparent HugePages (THP) feature is enabled by default in Oracle Linux. However, you might still need to access and configure THP according to the system's needs.

The following sections look at various THP parameters and examples of how they can be configured.

- Parameters Used to Configure Transparent HugePages
- Retrieving the Current Status of Transparent HugePages
- Changing the Current Status of Transparent HugePages
- Changing the defrag Setting of Transparent HugePages

Parameters Used to Configure Transparent HugePages

The following table describes selected parameter settings that can be used when configuring Transparent HugePages (THP).

Table 3-1 Commonly Used THP Parameters

Parameter	File Location	Value Options
`enabled`	`/sys/kernel/mm/transparent_hugepage/enabled`	Sets THP and its mode, which is one of the following: • `always` (default): THP is enabled in `system-wide` mode. In this setting, the kernel, whenever possible, assigns huge pages to processes using large contiguous virtual memory areas. • `madvise`: THP is enabled in `per-process` mode. In this setting the kernel only assigns huge pages to application processes that explicitly request huge pages through the `madvise()` system call. • `disabled`: THP is disabled.

Table 3-1 (Cont.) Commonly Used THP Parameters

Parameter	File Location	Value Options
`defrag`	`/sys/kernel/mm/transparent_hugepage/defrag`	Determines how aggressively an application can reclaim pages and defrag memory when THP is unavailable. The following list explains the available options: • `always`: An application requesting THP stalls on allocation failure and directly reclaims pages and compact memory to obtain a THP immediately. • `defer`: An application doesn't stall but continues using small pages. The application requests the kernel daemons `kswapd` and `kcompactd` to reclaim pages and compact memory so that THP is available later. • `defer+madvise`: Regions using the `madvise(MADV_HUGEPAGE)` call stall on allocation failure and directly reclaim pages and compact memory to obtain a THP immediately However, all other regions request the kernel daemons `kswapd` and `kcompactd` to reclaim pages and compact memory so that THP is available later. • `madvise` (default): Regions using the `madvise(MADV_HUGEPAGE)` call stall on allocation failure and directly reclaim pages and compact memory to obtain a THP immediately.

Retrieving the Current Status of Transparent HugePages

This procedure shows how to see the current status of THP in the `sysfs` virtual file system.

For more information about using the `sysfs` virtual file system, see Oracle Linux 9: Managing Kernels and System Boot

To see the current setting of THP you can read the `/sys/kernel/mm/transparent_hugepage/enabled` parameter.

- Use the `cat` command to view the current value of the THP in the `sysfs`.

```
cat /sys/kernel/mm/transparent_hugepage/enabled
```

```
[always] madvise never
```

The value inside the square brackets represents the current setting.

Changing the Current Status of Transparent HugePages

This procedure shows how to change the current status of THP by setting a value in the `sysfs` virtual file system.

For more information about using the `sysfs` virtual file system, see Oracle Linux 9: Managing Kernels and System Boot

To change the current status of THP, you need to write the preferred settings to `/sys/kernel/mm/transparent_hugepage/enabled`. The following example shows you how to set the status to `always`.

> **Note:**
>
> Virtual file systems such as `sysfs` provide a file system interface to items that aren't necessarily stored as files on disk. The `sysfs` files therefore don't always interact with file commands in the same way that regular physical files on disk would. In the example procedure, the `echo` command used doesn't overwrite `/sys/kernel/mm/transparent_hugepage/enabled`, as it would if used with a regular file, but instead changes the selected option.

1. Check the current status of THP by reading the `enabled` parameter.

```
sudo cat /sys/kernel/mm/transparent_hugepage/enabled
```

```
always madvise [never]
```

The value inside the square brackets represents the current setting.

2. Set THP mode to `always`.

```
echo always | sudo tee /sys/kernel/mm/transparent_hugepage/enabled
```

3. Confirm the change has been successful by reading the `enabled` parameter.

```
sudo cat /sys/kernel/mm/transparent_hugepage/enabled
```

```
[always] madvise never
```

Changing the defrag Setting of Transparent HugePages

This procedure shows how to change the defragmentation setting of THP by setting a value in the `sysfs` virtual file system.

For more information about using the `sysfs` virtual file system, see Oracle Linux 9: Managing Kernels and System Boot

To change the THP defrag setting you need to write the new setting to `/sys/kernel/mm/transparent_hugepage/defrag`

> **Note:**
>
> The best `defrag` setting varies from system to system. Reclaiming pages and memory compaction can increase the number of THP pages available. However, the process also uses CPU time. Therefore, you need to find the correct balance for a specific system.

The following example shows you how to set the `defrag` setting to `madvise`.

1. Check the current value of the `defrag` parameter:

```
sudo cat /sys/kernel/mm/transparent_hugepage/defrag
```

```
[always] defer defer+madvise madvise never
```

   The value inside square brackets represents the current setting.

2. Set the `/sys/kernel/mm/transparent_hugepage/defrag` parameter to `madvise`:

```
echo madvise | sudo tee /sys/kernel/mm/transparent_hugepage/defrag
```

3. Confirm the change by reading the `defrag` parameter.

```
sudo cat /sys/kernel/mm/transparent_hugepage/defrag
```

```
always defer defer+madvise [madvise] never
```