Oracle Linux: HugePages What It Is... and What It Is Not... (Doc ID 361323.1)

In this Document

APPLIES TO:

Gen 1 Exadata Cloud at Customer (Oracle Exadata Database Cloud Machine) - Version N/A and later
Oracle Cloud Infrastructure - Database Service - Version N/A and later
Oracle Database Exadata Express Cloud Service - Version N/A and later
Oracle Database Cloud Exadata Service - Version N/A and later
Oracle Database Cloud Schema Service - Version N/A and later
Linux x86
Linux on IBM Z
IBM S/390 Based Linux (31-bit)
IBM: Linux on POWER Big Endian Systems
Linux x86-64
Linux Itanium

PURPOSE

This document describes the HugePages feature in the Linux kernel available for 32-bit and 64-bit architectures. There has been some confusion among the terms and uses related to HugePages. This document should clarify the misconceptions about the feature.

SCOPE

Information in this document is useful for Linux system administrators and Oracle database administrators working with system administrators.

This document covers information about HugePages concept that applies to very large memory (VLM) (>= 4GB) systems for 32-bit and 64-bit architectures including some configuration information and references.

DETAILS

Introduction

HugePages is a feature integrated into the Linux kernel with release 2.6 and later. This feature basically provides the alternative to the 4K page size (16K for IA64) providing bigger pages.

Regarding the HugePages, there are some other similar terms that are being used like, hugetlb, hugetlbfs. Before proceeding into the details of HugePages, see the definitions below:

- **Page Table**: A page table is the data structure of a virtual memory system in an operating system to store the mapping between virtual addresses and physical addresses. This means that on a virtual memory system, the memory is accessed by first accessing a page table and then accessing the actual memory location implicitly.
- **TLB:** A Translation Lookaside Buffer (TLB) is a buffer (or cache) in a CPU that contains parts of the page table. This is a fixed size buffer being used to do virtual address translation faster.
- **hugetlb:** This is an entry in the TLB that points to a HugePage (a large/big page larger than regular 4K and predefined in size). HugePages are implemented via hugetlb entries, i.e. we can say that a HugePage is handled by a "hugetlb page entry". The 'hugetlb" term is also (and mostly) used synonymously with a HugePage (See Note 261889.1). In this document the term "HugePage" is going to be used but keep in mind that mostly "hugetlb" refers to the same concept.
- **hugetlbfs:** This is a new in-memory filesystem like tmpfs and is presented by 2.6 kernel. Pages allocated on hugetlbfs type filesystem are allocated in HugePages.

Common Misconceptions

WRONG: HugePages is a method to be able to use large SGA on 32-bit VLM systems	**RIGHT:** HugePages is a method to have larger pages where it is useful for working with very large memory. It is both useful in 32- and 64-bit configurations
WRONG: HugePages cannot be used without USE_INDIRECT_DATA_BUFFERS	**RIGHT:** HugePages can be used without indirect buffers. 64-bit systems do not need to use indirect buffers to have a large buffer cache for the RDBMS instance. Furthermore HugePages can be used with indirect buffers.
WRONG: hugetlbfs means hugetlb	**RIGHT:** hugetlbfs is a filesystem type **BUT** hugetlb is the mechanism employed in the back where hugetlb can be employed WITHOUT hugetlbfs
WRONG: hugetlbfs means hugepages	**RIGHT:** hugetlbfs is a filesystem type **BUT** HugePages is the mechanism employed in the back (synonymously with hugetlb) where HugePages can be employed WITHOUT hugetlbfs.

Regular Pages and HugePages

This section aims to give a general picture about memory access in virtual memory systems and how pages are referenced.

When a single process works with a piece of memory, the pages that the process uses are reference in a local page table for the specific process. The entries in this table also contain references to the System-Wide Page Table which actually has references to actual physical memory addresses. So theoretically a user mode process (i.e. Oracle processes), follows its local page table to access to the system page table and then can reference the actual physical table virtually. As you can see below, it is also possible (and very common to Oracle RDBMS due to SGA use) that two different O/S processes can point to the same entry in the system-wide page table.

When HugePages are in the play, the usual page tables are employed. The very basic difference is that the entries in both process page table and the system page table has attributes about huge pages. So any page in a page table can be a huge page or a regular page.

HugePages in 2.4 Kernels

The HugePages feature is backported to some 2.4 kernels. Kernel versions 2.4.21-* has this feature (See Note 311504.1 for the distributions with 2.4.21 kernels) but it is implemented in a different way. The feature is completely available. The difference from 2.6 implementation is the organization within the source code and the kernel parameters that are used for configuring HugePages. See Parameters/Setup section below.

Some HugePages Facts/Features

- HugePages can be allocated on-the-fly but they must be reserved during system startup. Otherwise the allocation might fail as the memory is already paged in 4K mostly.
- HugePage sizes vary from 2MB to 256MB based on kernel version and HW architecture (See related section below.)
- HugePages are not subject to reservation / release after the system startup unless there is system administrator intervention, basically changing the hugepages configuration (i.e. number of pages available or pool size)

Advantages of HugePages Over Normal Sharing Or AMM

- **Not swappable:** HugePages are not swappable. Therefore there is no page-in/page-out mechanism overhead.HugePages are universally regarded as pinned.
- **Relief of TLB pressure:**
  - Hugepge uses fewer pages to cover the physical address space, so the size of "book keeping" (mapping from the virtual to the physical address) decreases, so it requiring fewer entries in the TLB
  - TLB entries will cover a larger part of the address space when use HugePages, there will be fewer TLB misses before the entire or most of the SGA is mapped in the SGA
  - Fewer TLB entries for the SGA also means more for other parts of the address space
- **Decreased page table overhead:** Each page table entry can be as large as 64 bytes and if we are trying to handle 50GB of RAM, the pagetable will be approximately 800MB in size which is practically will not fit in 880MB size lowmem (in 2.4 kernels - the page table is not necessarily in lowmem in 2.6 kernels and later) considering the other uses of lowmem. When 95% of memory is accessed via 256MB hugepages, this can work with a page table of approximately 40MB in total. See also Document 361468.1.
- **Eliminated page table lookup overhead:** Since the pages are not subject to replacement, page table lookups are not required.
- **Faster overall memory performance:** On virtual memory systems each memory operation is actually two abstract memory operations. Since there are fewer pages to work on, the possible bottleneck on page table access is clearly avoided.

The Size of a HugePage

The size of a single HugePage varies according to:

- Kernel version/linux distribution
- HW Platform

The actual size of the HugePage on a specific system can be checked by:

```
$ grep Hugepagesize /proc/meminfo
```

The table below shows the sizes of HugePages on different configurations. Note that these are general numbers taken from the most recent versions of the kernels. For a specific kernel source package, you can check for the HPAGE_SIZE macro value (based on HPAGE_SHIFT) for a different (more recent) kernel source tree.

HW Platform	Source Code Tree	Kernel 2.4	Kernel 2.6 and later
Linux x86 (IA32)	i386	4 MB	2 MB
Linux x86-64 (AMD64, EM64T)	x86_64	2 MB	2 MB
Linux Itanium (IA64)	ia64	256 MB	256 MB
IBM Power Based Linux (PPC64)	ppc64/powerpc	N/A **	16 MB

IBM zSeries Based Linux	s390	N/A	1 MB
IBM S/390 Based Linux	s390	N/A	N/A

* Some older packaging for the 2.6.5 kernel on SLES8 (like 2.6.5-7.97) can have 2 MB Hugepagesize.
** Oracle RDBMS is also not certified in this configuration. See Document 341507.1

HugePages Reservation

The HugePages reservation feature is fully implemented in 2.6.17 kernel, and thus EL5 (based on 2.6.18) has this feature. The alloc_huge_page() is improved for this. (See kernel source mm/hugetlb.c)

From /usr/share/doc/kernel-doc-2.6.18/Documentation/vm/hugetlbpage.txt:

> HugePages_Rsvd is short for "reserved," and is the number of hugepages for which a commitment to allocate from the pool has been made, but no allocation has yet been made. It's vaguely analogous to overcommit.

This feature in the Linux kernel enables the Oracle Database to be able to allocate hugepages for the sublevels of the SGA on-demand. The same behaviour is expected for various Oracle Database versions that are certified on EL5.

HugePages and Oracle 11g Automatic Memory Management (AMM)

The AMM and HugePages are not compatible. One needs to disable AMM on 11g to be able to use HugePages. See Document 749851.1 for further information.

What if Not Enough HugePages Configured?

Configuring your Linux OS for HugePages is a delicate process where if you do not configure properly, the system may experience serious problems. If you do not have enough HugePages configured you may encounter:

- HugePages not used (HugePages_Total = HugePages_Free) at all wasting the amount configured for
- Poor database performance
- System running out of memory or excessive swapping
- Some or any database instance cannot be started
- Crucial system services failing (e.g.: CRS)

To avoid / help with such situations Bug 10153816 was filed to introduce a database initialization parameter in 11.2.0.2 (use_large_pages) to help manage which SGAs will use huge pages and potentially give warnings or not start up at all if they cannot get those pages.

What if Too Much HugePages Configured?

It is of course technically possible to configure more than needed. When that is done, the unused part of HugePages allocation will not be available for other purposes on the system and memory shortage can be encountered. Please make sure to configure only for needed amount of hugepages.

Parameters/Setup

The following configurations are a minimal list of documents providing general guidelines to configure HugePages for more than one Oracle RDBMS instance:

- Document 317055.1 How to Configure RHEL 3.0 32-bit for Very Large Memory with ramfs and hugepages
- Document 317067.1 How to Configure Asianux 1.0 32-bit for Very Large Memory with ramfs and hugepages
- Document 317141.1 How to Configure RHEL 4 32-bit for Very Large Memory with ramfs and hugepages
- Document 317139.1 How to Configure SuSE SLES 9 32-bit for Very Large Memory with ramfs and hugepages
- Document 361468.1 HugePages on 64-bit Linux

For all configurations be sure to have environment variable DISABLE_HUGETLBFS is unset. If it is set (specifically to 1) it will disable the use of HugePages by Oracle database.

The performed configuration is basically based on the RAM installed and combined size of SGA of database instances you are running. Based on that when:

- Amount of RAM installed for the Linux OS changed
- New database instance(s) introduced
- SGA size / configuration changed for one or more database instances

you should revise your HugePages configuration to make it suitable to the new memory framework. If not you may experience one or more problems below on the system:

- Poor database performance
- System running out of memory or excessive swapping
- Database instances cannot be started
- Crucial system services failing

Kernel Version 2.4

The kernel parameter used for HugePages is vm.hugetlb_pool which is based on MB of memory. RHEL3, Asianux 1.0, SLES8 (Service Pack 3 and over) are examples of distributions with the 2.4 kernels with HugePages support. For the configuration, follow steps below:

1. Start database instance(s)
2. Calculate hugetlb_pool using script from Note 401749.1
3. Shutdown database instances
4. Set kernel parameter:
   # sysctl -w vm.hugetlb_pool=<value from above>
   and make sure that the parameter is persistent to reboots. e.g. On Asianux 1.0 by editing /etc/sysctl.conf adding/modifying as below:
   vm.hugetlb_pool=<value from above>

5. Check available hugepages:
   $ grep Huge /proc/meminfo

6. Restart database instances
7. Check available hugepages:
   $ grep Huge /proc/meminfo

Notes:

- If the setting of hugetlb_pool is not effective, you will need to reboot the server to make HugePages allocation during system startup.
- The HugePages are allocated in a lazy fashion, so the "Hugepages_Free" count drops as the pages get touched and are backed by physical memory. The idea is that it's more efficient in the sense that you don't use memory you don't touch.
- If you had set the instance initialization parameter PRE_PAGE_SGA=TRUE (for suitable settings see Document 30793.1), all of the pages would be allocated from HugePages up front.

Kernel Version 2.6 and later

The kernel parameter used for HugePages is vm.nr_hugepages which is based on the number of the pages. SLES9, RHEL4 and Asianux 2.0 are examples of distributions with the 2.6 kernel. For the configuration, follow steps below:

1. Start database instance(s)
2. Calculate nr_hugepages using script from Document 401749.1
3. Shutdown database instances
4. Set kernel parameter:
    # sysctl -w vm.nr_hugepages=<value from above>
and make sure that the parameter is persistent to reboots. e.g. On SLES9:
    # chkconfig boot.sysctl on

5. Check available hugepages:
    $ grep Huge /proc/meminfo

6. Restart database instances
7. Check available hugepages:
    $ grep Huge /proc/meminfo

Notes:

- If the setting of nr_hugepages is not effective, you will need to reboot the server to make HugePages allocation during system startup.
- The HugePages are allocated in a lazy fashion, so the "Hugepages_Free" count drops as the pages get touched and are backed by physical memory. The idea is that it's more efficient in the sense that you don't use memory you don't touch.
- If you had set the instance initialization parameter PRE_PAGE_SGA=TRUE (for suitable settings see Document 30793.1), all of the pages would be allocated from HugePages up front.

Notes on HugePages in General

- The userspace application that employs HugePages should be aware of permission implications. Permissions HugePages segments in memory can strictly impose certain requirements. e.g. Per Bug 6620371 on Linux x86-64 port of Oracle RDBMS until 11g was setting the shared memory flags to hugetlb, read and write by default. But that shall depend on the configuration environment and with Patch 6620371 on 10.2 and with 11g, the read and write permissions are set based on the internal context.

For RedHat 6, Oracle Linux 6, SLES 11 and UEK2 kernels please have at "ALERT: Disable Transparent HugePages on SLES11, RHEL6, Oracle Linux 6 and UEK2 Kernels (Doc ID 1557478.1)"

REFERENCES

NOTE:401749.1 - Oracle Linux: Shell Script to Calculate Values Recommended Linux HugePages / HugeTLB Configuration

BUG:10153816 - WHEN USE_LARGE_PAGES=ONLY AND NO HUGEPAGES EXIST STARTUP FAILS NO DIAGNOSTIC

NOTE:317139.1 - How to Configure SuSE SLES 9 / 10 32-bit for Very Large Memory with ramfs and HugePages
NOTE:311504.1 - QREF: Linux Kernel Version Nomenclature

NOTE:317067.1 - How to Configure Asianux 1.0 32-bit for Very Large Memory with ramfs and hugepages
NOTE:452326.1 - Linux Kernel Lowmem Pressure Issues and Related Kernel Structures