debian

Why huge pages?

When a process uses some memory, the CPU is marking the RAM as used by that process. For efficiency, the CPU allocate RAM by chunks of 4K bytes (it's the default value on many platforms). Those chunks are named *pages*. Those pages can be swapped to disk, etc.

Since the process address space are virtual, the CPU and the operating system have to remember which page belong to which process, and where it is stored. Obviously, the more pages you have, the more time it takes to find where the memory is mapped. When a process uses 1GB of memory, that's 262144 entries to look up (1GB / 4K). If one Page Table Entry consume 8bytes, that's 2MB (262144 * 8) to look-up.

Most current CPU architectures support bigger pages (so the CPU/OS have less entries to look-up), those are named *Huge pages* (on Linux), *Super Pages* (on BSD) or *Large Pages* (on Windows), but it all the same thing.

Contents

Read the documentation for more information about hugetlbpage.

Enabling HugeTlbPage

Currently, there is no standard way to enable HugeTLBfs, mainly because the FHS has no provision for such kind of virtual file system, see Open in sysvinit/2.87dsf-8.1, sysvinit/2.88dsf-13.13, sysvinit/2.88dsf-57: #572733: support for mounting other kernel filesystems: 572733 . (Fedora mounts it in `/dev/hugepages/`, so don't be surprised if you find some example on the web that use this location)

Linux support "Huge page tables" (HugeTlb) is available in Debian since DebianLenny (actually, since DebianBug: 2.6.23). A good introduction to large pages is available from ibm.com.

1. Create a group for users of hugepages, and retrieve it's GID (is this example, 2021) then add yourself to the group.
   Note: this should not be needed for libvirt (see `/etc/libvirt/qemu.conf`)

```
% groupadd my-hugetlbfs


% getent group my-hugetlbfs
```

```
my-hugetlbfs:x:2021:

% adduser franklin my-hugetlbfs
Adding user `franklin' to group `my-hugetlbfs' ...
Adding user franklin to group my-hugetlbfs
Done.
```

2. Edit `/etc/sysctl.conf` and add this text to specify the number of pages you want to reserve (see pages-size)

```
# Allocate 256*2MiB for HugePageTables (YMMV)
vm.nr_hugepages = 256

# Members of group my-hugetlbfs(2021) can allocate "
vm.hugetlb_shm_group = 2021
```

3. Create a mount point for the file system

```
% mkdir /hugepages
```

4. Add this line in `/etc/fstab` (The mode of 1770 allows anyone in the group to create files but not unlink or rename each other's files.[1])

```
hugetlbfs /hugepages hugetlbfs mode=1770,gid=2021 0
```

5. Reboot (This is the most reliable method of allocating huge pages before the memory gets fragmented. You don't necessarily *have to* reboot. You can try to run `sysctl -p` to apply the changes. if

`grep "Huge" /proc/meminfo` don't show all the pages, you can try to free the cache with `sync ; echo 3 > /proc/sys/vm/drop_caches` (where "3" stands for "purge pagecache, dentries and inodes") then try `sysctl -p` again. [2])

limits.conf

You should configure the amount of memory a user can lock, so an application can't crash your operating system by locking all the memory. Note that any page can be locked in RAM, not just huge pages. You should allow the process to lock a little bit more memory that just the the space for hugepages.

```
## Get huge-page size:
% grep "Hugepagesize:" /proc/meminfo
Hugepagesize:       4096 kB

## What's the current limit
% ulimit -H -l
64

## Just add them up... (how many pages do you want to a
```

See Limits (`ulimit -l` and `memlock` in `/etc/security/limits.conf`).

Multiple huge page size support

Some architectures (like ia64) can have multiple and/or configuration "huge" pages size.

(TODO)

See:

- boot parameters and mount options in hugetlbpage.txt in documentations.

arm64

The Debian arm64 kernel (running with a 4KB standard PAGE_SIZE) supports 2MB and 1GB HugeTLB page sizes. One has to pre-allocate 1GB HugeTLB pages on boot by specifying arguments on the kernel command line, the following will pre-allocate 10 x 1GB huge pages:

```
hugepagesz=1G hugepages=10
```

If one elects to build their own Debian arm64 kernel with CONFIG_ARM64_64K_PAGES=y, then only 512MB HugeTLB (and THP) pages are available. These are available at run time.

x86_64

Depending on the processor, there are at least two different huge page sizes on the x86_64 architecture: 2MB and 1GB. If the CPU supports 2MB pages, it has the PSE cpuinfo flag, for 1GB it has the PDPE1GB flag. `/proc/cpuinfo` shows whether the two flags are set.

If this commands returns a non-empty string, 2MB pages are supported.

```
% grep pse /proc/cpuinfo | uniq
flags           : [...] pse [...]
```

If this commands returns a non-empty string, 1GB pages are supported.

```
% grep pdpe1gb /proc/cpuinfo | uniq
flags           : [...] pdpe1gb [...]
```

Before they are actually both available, they may have to be activated at boot time. The following kernel boot parameters enable 1GB pages and create a pool of one 1GB page:

```
hugepagesz=1GB hugepages=1
```

After boot, the huge page pools look like this:

```
% hugeadm --pool-list
      Size  Minimum  Current  Maximum  Default
   2097152        0        0        0        *
1073741824        1        1        1
```

Getting informations

You can get a list of available huge page sizes with `hugeadm`:

```
% hugeadm --page-sizes-all
2097152
1073741824
```

`hugeadm` also displays the number of allocated huge pages per available size:

```
% hugeadm --pool-list
      Size  Minimum  Current  Maximum  Default
   2097152        0        0        0        *
1073741824        1        1        1
```

An alternative to retrieve the current available/used page for the default huge page size is `/proc/meminfo`:

```
% grep Huge /proc/meminfo
HugePages_Total:     256
HugePages_Free:      256
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       4096 kB
```

(read `Documentation/vm/hugetlbpage.txt` for more information about it)

Standard Debian Kernel have HUGETLB enabled (What about Lenny ? Xen ?):

```
% grep HUGETLB /boot/config-$(uname -r)
CONFIG_HUGETLBFS=y
CONFIG_HUGETLB_PAGE=y
```

Various runtime settings (see documentation)

```
% grep -R "" /sys/kernel/mm/hugepages/ /proc/sys/vm/
/sys/kernel/mm/hugepages/hugepages-4096kB/nr_hugepag
```

```
/sys/kernel/mm/hugepages/hugepages-4096kB/nr_overcom
/sys/kernel/mm/hugepages/hugepages-4096kB/free_hugep
/sys/kernel/mm/hugepages/hugepages-4096kB/resv_hugep
/sys/kernel/mm/hugepages/hugepages-4096kB/surplus_hu
/proc/sys/vm/hugepages_treat_as_movable:0
/proc/sys/vm/hugetlb_shm_group:0
/proc/sys/vm/nr_hugepages:256
/proc/sys/vm/nr_overcommit_hugepages:0
```

Huge pages sizes

Architecture	**huge page size**
arm64	4K, 2M and 1G (or 64K and 512M if one builds their own kernel with CONFIG_ARM64_64K_PAGES=y)
i386	4K and 4M (2M in PAE mode)
ia64	4K, 8K, 64K, 256K, 1M, 4M, 16M, 256M
ppc64	4K and 16M

Tools

- libhugetlbfs is a library which provides easy access to huge pages of memory. The library also comes with several userspace tools to help with huge page usability, environment setup, and control.
  http://libhugetlbfs.ozlabs.org/
  DebianBug: 533708 - ITP: libhugetlbfs -- Initial package request
  http://www.ibm.com/developerworks/wikis/display/LinuxP/libhuge+short+and+simple
- HugeTLB for PostgreSQL (and other apps) non-official
  http://oss.linbit.com/hugetlb/

Hugepage enabled applications

An application can allocate/use HugeTlbPage through two different means:

1. `mmap` system call require a mounted `hugetlbfs`, with appropriate permissions.
2. Shared memory segment (shmat/shmget system calls or mmap with MAP_HUGETLB), must be member of a group, configured in `/proc/sys/vm/hugetlb_shm_group`.

Application	**hugetlbfs**	**shared memory**
QEMU/KVM	Yes	No
MySQL	No	Yes
Java	No	Yes

MySQL

(TODO), see:

- Enabling Large Page Support http://dev.mysql.com/doc/refman/5.1/en/large-page-support.html

Linux HugeTLBfs: Improve MySQL Database Application Performance http://www.cyberciti.biz/tips/linux-hugetlbfs-and-mysql-performance.html (note tested)

Java (Sun, OpenJDK)

Sun and OpenJDK can use large pages.

- DebianPkg: sun-java6-bin - Sun Java(TM) Runtime Environment (JRE) 6 (architecture dependent files)
- DebianPkg: openjdk-6-jre-headless - OpenJDK Java runtime, using Hotspot JIT (headless)

  (TODO)

Basically, it seems that one have to launch java with

-XX:+UseLargePages. The maximum size of the Java heap (-Xmx) should fit in your reserved Huge pages ; same for `ulimit -l` and/or `memlock` in `/etc/security/limits.conf`.

See:

- Java Support for Large Memory Pages - Sun http://java.sun.com/javase/technologies/hotspot/largememory.jsp
- Configuring large page memory allocation - IBM User Guides for Java v6 on Linux http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp?topic=/com.ibm.java.doc.user.lnx.60/user/alloc_large_page.html

Possible errors:

- Insufficient `/proc/sys/kernel/shmmax` ?
- Not in group defined in `/proc/sys/vm/hugetlb_shm_group`

```
% java -XX:+UseLargePages
Java HotSpot(TM) Server VM warning: Failed to reserv
Java HotSpot(TM) Server VM warning: Failed to reserv
```

Memcached

DebianPkg: memcached can use huge pages, read the manpage :

memcached -L

Try to use large memory pages (if available). Increasing the memory page size could reduce the number of TLB misses and improve the performance. In order to get large pages from the OS, memcached will allocate the total item-cache in one large chunk. Only available if supported *on your OS.*

PosgreSQL

PostgreSQL gains huge page support in version 9.4. For previous versions, see hugetlblib above.

Virtualisation

Some consideration about hugepages an virtualisation.

1. Before enabling hugepages in a virtual machine, you should make sure the that your virtualization tool can handle it.
2. Whether a virtualization tools supports hugepages for it's client or for itself are probably two different aspects.

KVM

(TODO), see:

- http://fedoraproject.org/wiki/Features/KVM_Huge_Page_Backed_Memory
- Get a performance boost by backing your KVM guest with hugetlbfs http://www.linux-kvm.com/content/get-performance-boost-backing-your-kvm-guest-hugetlbfs (not tested)

Xen

(TODO)

It is unclear which version of Xen supports huge pages, and how it is used.

- See http://zhigang.org/wiki/XenHugePages

Documentation

- Documentation http://git.kernel.org/....../Documentation/vm/hugetlbpage.txt
- DebianMan: alloc_hugepages(2), DebianMan: free_hugepages(2) - allocate or free huge pages
- http://www.fogproject.org/wiki/index.php?title=Kernel_Parameters - Boot options for huge page sizes and pools
- http://lwn.net/Articles/376606/ - Article about managing huge pages

See also

- Article about Huge pages, on LWN.net Introduction - Interfaces - Administration - Benchmarking

1. Credits: Russell Coker (1)
2. Credits for *drop_caches*: http://oss.linbit.com/hugetlb/ (2)