This website uses cookies to ensure you get the best experience on our website

OKAY MORE INFO

« Red Hat / Oracle Enterprise Linux I/O tuning for Oracle databases

Veritas File System (VxFS) tuning

Linux Huge Pages and virtual memory (VM) tuning

Table Of Contents

Preamble

With huge hardware prices decrease and all consolidation/virtualization projects going around your Oracle database servers are most probably quite powerful and running multiple instances. The ones I'm using are made of two X7560 x86 processors wit 8 cores each (so 16 cores and you even see 32 CPUs if Hyper Threading is activated) and 64 GB RAM. My servers are running Red Hat Enterprise Linux Server release 5.5 (Tikanga) (Red Hat 6 not certified at the tir we installed servers) and Oracle 11gR2 and 10gR2 (test database of this post is Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 – 64bit Production).

The bigger number of databases per server increases complexity and obviously requires more expertise to understan if a database performance issue is linked to Oracle or server or another instance runing on same server and eating all resources. The performance problem we have experienced was linked to server but reported by one application only: database queries of this application very slow (AWR reports automatically canceled) and/or databases not even reachable. This, even if allocated Oracle memory for all instances was far below maximum. CPU usage on server whe experiencing slow response time was also low.

What happens in the background is very well explained is My Oracle Support (MOS) note 361670.1 and in Pythian po called Performance tuning: Huge Pages in Linux (see references section).

Obviously I have experienced it on my Linux box:

```
[root@server1 ~]# vmstat -s |grep paged
 159809376876 pages paged in
  12513585830 pages paged out
[root@server1 ~]# vmstat -s |grep paged
 159814312328 pages paged in
  12513708731 pages paged out
```

This website uses cookies to ensure you get the best experience on our website OKAY MORE INFO

```
[root@server1 ~]# sar -r -t /var/log/sa/sa09
Linux 2.6.18-274.3.1.el5 (server1)      07/09/12

00:00:01    kbmemfree kbmemused  %memused kbbuffers  kbcached kbswpfree kbswpused  %swpused  kb
.
.
12:50:01     54859152  11002672     16.71     46424   7517192  43599680   5912760     11.94
13:00:02     57649756   8212068     12.47     10492   4952724  40570716   8941724     18.06
.
.
21:10:01     52864804  12997020     19.73    278332   8989132  41647328   7865112     15.89
21:20:01     35866680  29995144     45.54    280840  25883808  41654116   7858324     15.87
.
.
22:00:01     35516316  30345508     46.07    289664  26246756  41698240   7814200     15.78
22:10:01     32650008  33211816     50.43    291904  29011612  43184728   6327712     12.78
.
.
```

And lots of swap in/out pages:

```
[root@server1 ~]# export LANG=C
[root@server1 ~]# sar -W -f /var/log/sa/sa09
Linux 2.6.18-274.3.1.el5 (server1)      07/09/12

00:00:01     pswpin/s pswpout/s
.
.
12:50:01         1.60      0.00
13:00:02        19.54   1204.79
.
.
21:10:01         6.83      0.00
21:20:01         4.17      0.00
.
.
22:00:01         6.70      0.00
22:10:01       627.21      0.00
.
.
```

So then digging around you will come to Linux Huge Pages which have the following benefit (please refer to documen in references section):

- Bigger page size (2MB on Linux x86_64) so reduced Page Table (512 times smaller: 2MB/4KB), reduced Translation Lookaside Buffer (TLB) and then less contention/CPU usage.
- Not swappable.

The only cons I could see is its non-flexibility means you may need to reboot your server to allocate Huge Pages whic can happen if you add a new database to your server and/or want to change current memory allocated for an already running database.

This website uses cookies to ensure you get the best experience on our website OKAY MORE INFO

```
HugePages_Free:       0
HugePages_Rsvd:       0
Hugepagesize:     2048 kB
```

> HugePages_Rsvd
> is short for “reserved,” and is the number of huge pages for which a commitment to allocate from the pool has been made, but no allocation has yet been made. Reserved huge pages guarantee that an application will be able to allocate a huge page from the pool of huge pages at fault time.

Page table size can be obtained with:

```
[root@server1 ~]# cat /proc/meminfo | grep PageTables
PageTables:    2137628 kB
```

More information on Huge Pages can be found by installing kernel-uek-doc-2.6.39-200.29.1.el6uek.noarch on Oracle Linux Server release 6.3:

```
[root@server1 ~]# cat /usr/share/doc/kernel-doc-2.6.39/Documentation/vm/hugetlbpage.txt
```

Huge Pages is **NOT** compatible with new 11g Automatic Memory Management (AMM) i.e. memory_max_target and memory_target so you have to use Automatic Shared Memory Management (ASMM) i.e. sga_max_size and sga_target.

Clearly a **regression** on this, this may change with further kernel releases, so it deserves a quick bench of your syste to understand if performance gain is worth flexibility lost.

Remark:
I have ever wonder the added value of SGA_TARGET versus SGA_MAX_SIZE as on all OS where I have tested it even if SGA_TARGET is much lower than SGA_MAX_SIZE the memory is anyway requested and allocated at OS lev (ipcs -m) so in clear useless functionality. Apparently this is not true on Solaris and on this OS the behavior is the one you can expect…

Huge Pages implementation

First get the gid of your Unix dba group:

```
[root@server1 ~]# grep dba /etc/group
dba:x:501:oracle
[root@server1 ~]# id -g oracle
501
```

Set this number to **vm.hugetlb_shm_group** kernel parameter:

```
[root@server1 ~]# sysctl -w vm.hugetlb_shm_group=501
vm.hugetlb_shm_group = 501
```

Change “The maximum size that may be locked into memory”, value that can be checked with:

This website uses cookies to ensure you get the best experience on our website OKAY MORE INFO

memory, 64GB in my case):

```
[root@server1 ~]# tail -2 /etc/security/limits.conf
oradmspoc   soft   memlock    60397977
oradmspoc   hard   memlock    60397977
```

Remark:
Even if it looks more secure to put account name. In a consolidated environment and/or when databases can switch from one server to another it could becomes complex to handle. So huge temptation to replace account name (i.e. oradmspoc) by * character.

As your Oracle Unix accounts are most probably in same Unix group (dba for example) a safer solution could be:

```
@dba soft memlock unlimited
@dba hard memlock unlimited
```

Logoff/logon and check it is active:

```
[oradmspoc@server1 ~]$ ulimit -l
60397977
```

Finally third parameter to change is **vm.nr_hugepages**, even if dynamic parameter you may encounter difficulties to change it and you may need to reboot your server (to change its value kernel must find contiguous free space and if you server is running since long memory is most probably quite fragmented). I have partially resolved it by submitting the kernel change multiple time. Please note it's a number of pages (2MB) and not a size in bytes:

```
[root@server1 ~]# echo 252 > /proc/sys/vm/nr_hugepages
[root@server1 ~]# sysctl -w vm.nr_hugepages=252
vm.nr_hugepages = 252
[root@server1 ~]# for i in $(seq 1 10); do echo 252 > /proc/sys/vm/nr_hugepages; sleep 10; done
```

Control it is effective:

```
[root@server1 ~]# sysctl vm.nr_hugepages
vm.nr_hugepages = 252
[root@server1 ~]# cat /proc/sys/vm/nr_hugepages
252
[root@server1 ~]# cat /proc/meminfo | grep HugePages_Total
HugePages_Total:   252
```

To which value set vm.nr_hugepages kernel parameter ? If your Oracle database are already running you may use Oracle script (MOS note 401749.1):

```
[root@server1 ~]# /home/oradmspoc/yannick/hugepages_settings.sh

This script is provided by Doc ID 401749.1 from My Oracle Support
(http://support.oracle.com) where it is intended to compute values for
the recommended HugePages/HugeTLB configuration for the current shared
memory segments. Before proceeding with the execution please make sure
that:
 * Oracle Database instance(s) are up and running
```

This website uses cookies to ensure you get the best experience on our website OKAY MORE INFO

```
Press Enter to proceed...

Recommended setting: vm.nr_hugepages = 252
```

Or allocate your (SGA size in MB/2)+1 pages (sum for all your Oracle database), take care of rounding on SGA i.e. allocation a SGA of exactly 401MB is not possible. I take the opportunity to write that RedHat formula I have seen in multiple document i.e. (SGA+PGA+(20KB * # of Oracle processes running)) / 2MB is wrong as Huge Pages is **ONLY** f SGA and not for PGA. If you really don't know how much databases will be finally running on your server Red Hat generic recommendation is **to set Huge Pages size to half your server physical memory**.

You can control usage with (or using **nmon**):

```
[root@server1 ~]# watch -n 10 cat /proc/meminfo
[root@server1 ~]# cat /proc/meminfo | grep Huge
```

Remark:
The real number of free Huge Pages is HugePages_Free – HugePages_Rsvd, if you want Oracle to initialize all Huge Pages use initialization parameter **pre_page_sga=true**.

Remark:
Starting with 11.2.0.2 there is a new initialization parameter called **use_large_pages** than can forbid Oracle to start (only value) if no Huge Pages are available. This avoid mistake and Oracle using normal page size (4KB).

Not swapping PGA

As we have seen above this Huge Pages story is only true for SGA parameter. But, obviously, we also want **NO** swapping for PGA memory so a bit of tuning on virtual memory (VM) subsystem is needed.

More information on VM kernel parameters can be found by installing kernel-uek-doc-2.6.39-200.29.1.el6uek.noarch o Oracle Linux Server release 6.3:

```
[root@server1 ~]# cat /usr/share/doc/kernel-doc-2.6.39/Documentation/sysctl/vm.txt
```

There are a lot of available RedHat documentation on the subject, they more or less all suggest the same parameter values except for vm.swappiness. Keeping in mind that on an Oracle server I rate swapping really bad I think setting it to 0 is a good idea:

Kernel Parameter	Description	Default Value	Recommended Value
vm.swappiness	This control is used to define how aggressive the kernel will swap memory pages. Higher values will increase aggressiveness, lower values decrease the amount of swap.	60	0
vm.dirty_background_ratio	Contains, as a percentage of total system memory, the number of pages at which the pdflush background writeback daemon will start writing out dirty data.	10	3

This website uses cookies to ensure you get the best experience on our website OKAY MORE INFO

vm.dirty_expire_centisecs	This tunable is used to define when dirty data is old enough to be eligible for writeout by the pdflush daemons. It is expressed in 100'ths of a second. Data which has been dirty in-memory for longer than this interval will be written out next time a pdflush daemon wakes up.	3000	500
vm.dirty_writeback_centisecs	The pdflush writeback daemons will periodically wake up and write `old' data out to disk. This tunable expresses the interval between those wakeups, in 100'ths of a second. Setting this to zero disables periodic writeback altogether.	500	100

References

- Slow Performance with High CPU Usage on 64-bit Linux with Large SGA [ID 361670.1]
- **Performance tuning: HugePages in Linux**
- HugePages on Linux: What It Is… and What It Is Not… [ID 361323.1]
- HugePages on Oracle Linux 64-bit [ID 361468.1]
- Shell Script to Calculate Values Recommended Linux HugePages / HugeTLB Configuration [ID 401749.1]
- USE_LARGE_PAGES To Enable HugePages In 11.2 [ID 1392497.1]
- ASMM and LINUX x86-64 Hugepages Support [ID 1134002.1]
- HugePages and Oracle Database 11g Automatic Memory Management (AMM) on Linux [ID 749851.1]
- Oracle Not Utilizing Hugepages [ID 803238.1]
- **Configuring Linux Hugepages for Oracle Database Is Just Too Difficult! Isn't It? Part – I.**
- **Huge Pages, Linux et Oracle (French)**
- **Performance tuning: HugePages in Linux**
- **Pythian Goodies: The Answer to Free Memory, Swap, Oracle, and Everything**
- **Thread: SGA_MAX_SIZE != SGA_TARGET when?**
- **How to Configure x86 Memory Performance for Large Databases**

Share the knowledge!

1

You may also like:

Uma mãe do Rio Database Information Médicos SQL Plan Pílula capaz de

This website uses cookies to ensure you get the best experience on our website

OKAY MORE INFO

segredo
Health News Online

OR usage

comprimido...
SaudeEFitness.co

lançada no Brasil
Health News Onlin

About the Author | Latest Posts

About Yannick Jaquier

Find more about me on:

Download article as PDF

Published: 12th of July 2012, 12:32 | Last updated: 8th of July 2015, 14:42 | Written by **Yannick Jaquier** | Tags: Automatic Memo Management (AMM), Automatic Shared Memory Management (ASMM), Huge Pages, Tuning, Virtual Memory | Category: Linux, Oracle

4 comments to Linux Huge Pages and virtual memory (VM) tuning

sulimo
December 11, 2012 at 22:14 · Reply

Nice article. Could you please explain in further detail the reason why PGA would not use huge pages?

Yannick Jaquier
January 29, 2013 at 12:47 · Reply

Thanks for comment ! 🙂

If you refer to Huge Page kernel documentation you can read:
http://www.kernel.org/doc/Documentation/vm/hugetlbpage.txt

This website uses cookies to ensure you get the best experience on our website OKAY MORE INFO

Use of mmap() system call requires a filesystem mounted with something like (never seen it in action so far):

```
mount -t hugetlbfs -o uid=110,gid=106,size=1M none /mnt/huge
```

As PGA is private memory I think it explains why it cannot benefit from Huge Pages.

This may change in RHEL 6 with Transparent Huge Pages (THP)…

Yannick.

Tmpfs vs Ramfs vs (Transparent) Huge Pages - IT World
January 14, 2014 at 13:06 · Reply

[…] we have seen in this blog post Huge Pages is an answer to swapping. But unfortunately as clearly mentioned in documentation […]

daVikes
January 14, 2016 at 15:00 · Reply

One thing covered is kernel.shmmax and kernel.shmmall and how that affects pga ? Also, anyone know of any good tools / scripts to see what memory is loaded in Huge Pages or monitor what is using all of your memory including Huge Pages? Trying to come up with these settings and get SGA / PGA / with Huge Pages and avoid swapping.

Thanks!