From: Spitzer, Andras ([redacted])
Sent: Friday, October 12, 2012 1:12 PM
To: [redacted] ([redacted])
Cc: [redacted] ([redacted]) ([redacted])
Subject: RE: VM SWAP
Attachments: swapp_info.py

Nate,

I promised to get back to you in this case, here it is. First of all I will talk a little about the one liner, second I will explain how disk swap works in detail within Solaris, so you'll see the proof your zone works without disk swap.

Your question was **whether I have a one liner of finding who is consuming swap**. I had no such one liner, but I tried to write one, I was curious myself.

First of all, **some background info about how swapping works** in Solaris. I already described in my previous e-mail why can be confusing talking about swap in Solaris, and why is it very important to differentiate disk swap and memory swap. Just to recap, in Solaris we call swap the physical memory + disk swap all together, and whenever a process starts it will see a virtual memory. Whenever the process starts to use that virtual memory area (read/write), the Solaris will automatically and transparently assigns a page sized physical device to back that memory request. The device can be physical memory or physical disk, the process will never be able to tell the difference. This is done through Solaris' segment drivers, where the process is actually writing to a segment, and Solaris is responsible to assign a segment driver to the request, which can be any device, but usually physical memory or physical disk. This way Solaris can also "map" areas from a graphics card (frame buffer) for example directly into a process' virtual memory.

Also it's important to know that Solaris will always prefer to use physical memory over physical disk. Now, about the disk swapping. Solaris by default will never write to the disk swap, until we have free available memory. More accurately until 'freemem > lotsfree' (kstat –n system_pages). When we drop low the available memory in our system, there is a pageout process kicks in (process id 2, although it's not a process it's part of the kernel), which starts scanning the memory for least used pages (LRU algorithm), and starts to migrate those pages to the physical disk. This is called demand paging, and the unit is memory pages (8192 bytes by default). (Btw you can read the pageout function here : http://src.opensolaris.org/source/xref/onnv/onnv-gate/usr/src/uts/common/os/vm_pageout.c#652)

This is the first time Solaris will ever start using disk swap. The second type of disk swapping is what we call "swap out", when not pages are moved to the disk swap area, but whole processes, although as this will cause huge performance degradation Solaris will start this process only after demand paging seems to be useless for a while (30+ seconds if I remember correctly), you can consider it as a more drastic approach when the smart approach seems not to be efficient enough for some reason. So swap outs you should never experience, only demand paging, but demand paging is rare as well. Very rare. Both cases you can monitor paging/swap outs with the 'sr' column in vmstat, stands for Scan Rate and which shows you the scan rate of the pageout process described earlier. If 'sr' is 0, you don't have disk swap activity. This is the simple most effective way to check. We talked about how pages are going out to disk swap, now about a few words how pages are getting back to memory. They are brought back to memory by a trap signal within the kernel. So the process is trying to access it's segment as it has no idea Solaris moved its page beneath from physical memory to disk swap, and as the segment drivers see the page is missing from memory it generates a trap signal, which will move the page back into memory, and after the trap completes the process can continue, without ever recognizing there was a page transfer (from disk to memory) in the background.

Now **about the one liners**, which process is consuming the swap.. First of all, as DTrace can attach to functions entry and exit points, we only can see what is happening, whether a process is paging in or paging out, we are unable to review the kernel who is using the disk swap at this point in time. Second, as we decided to find out who is paging in and paging out to disk swap, actually only one of these is possible. We can tell which process is paged in from disk swap, but there is no way we can tell which process is paged out to disk swap. Why? It's simple. Because the thread which initiates page outs to disk swap is a kernel thread and neither knows nor cares about processes. I've read through the function, there is not even a single back pointer to which process the page relates to. So the pageout process scans the physical memory and only cares about pages, it doesn't care about where which process(es) they

belong to. Also as it's initiated and run by a kernel thread, with DTrace I only can see that it was initiated by "pageout", PID 2.

To illustrate the difference between page in and page out to disk swap, let me show you the function stack of both, you'll see the page out is initiated by the kernel thread while page in is by the process' trap signal.

Page in from disk swap (handled by a kernel trap signal initiated by the user thread in the process which was trying to access that page):

```
genunix`swap_getapage+0x26c
genunix`swap_getpage+0x4c
genunix`fop_getpage+0x44
genunix`anon_getpage+0xfc
genunix`anon_map_getpages+0xe0
genunix`segvn_fault_anonpages+0x32c
genunix`segvn_fault+0x530
genunix`as_fault+0x4c8
unix`pagefault+0xac
unix`trap+0xd50
unix`utl0+0x4c
```

Page out to disk swap (initiated by kernel thread):

```
genunix`swap_newphysname+0x2c
genunix`swap_putapage+0x218
genunix`swap_putpage+0x22c
genunix`fop_putpage+0x1c
genunix`pageout+0x1d4
unix`thread_start+0x4
```

So we can't tell which process is paged out using DTrace. Although we can tell which process pages in. The script looks like this :

```
#!/usr/sbin/dtrace -qs

fbt::swap_phys_free:entry
{
        printf("%Y exec: %s pid: %d uid: %d size: %d maj : %d min : %d\n",
walltimestamp, execname, pid, uid,
          args[2], (args[0]->v_rdev >> 32) & 0xffffffffl, args[0]->v_rdev &
0xffffffffl );
}
```

We attach to swap_phys_free, which is a function to copy a page back into physical memory, and as it's initiated by the process the page belongs to, DTrace can show us who had their memory paged out (before they called this function as the function's goal is to move the page back into memory and release disk swap). An example of the output :

```
2012 Oct  8 22:41:24 exec: sendmail pid: 9401 uid: 25 swap_phys_free entry size: 8192
maj : 256 min : 1
2012 Oct  8 22:41:24 exec: sendmail pid: 9401 uid: 25 swap_phys_free entry size: 8192
maj : 256 min : 1
2012 Oct  8 22:42:00 exec: svc.configd pid: 11 uid: 0 swap_phys_free entry size: 8192
maj : 256 min : 1
```

You see the size is a single page (8k) and the major / minor numbers are pointing to the swap device (256,1) :

```
gislabsol04# /usr/sbin/swap -l
swapfile                 dev        swaplo   blocks   free
/dev/zvol/dsk/rpool/swap  256,1      16       16777200 16552784
```

After I got these results, I was still thinking about how could I show you what processes are using disk swap within a system. Then I realized I can use mdb to scan the current kernel using walkers in mdb. I had to resolve all the paths and structures from segments to anon pages (used the Solaris 10 Internals book and the Opensolaris source code), and it went pretty good until I got to a point where I had to dereference pointers of pointers (**array_chunk). That was the point I couldn't go further with a single mdb command, so I had to write a wrapper script which controls mdb. That was fun, and I ended up writing a script called swap_info.py (attached). Please if you want to run it run it only on dev or test environment as I wrote it only to prove a point without any safety checks, but using the script finally I was able to resolve which 'processes are having segments that are backed by vnodes resides on one of the swap devices'. I had many problems during the script, for example as mdb realize I'm not in interactive mode it won't give me a prompt, so I had to come up with an idea how to find out if a command finished or not, etc. But in short it works.

It goes through all the processes within mdb and their segments and will look for anon pages backed by vnodes on the swap devices. If you count the actual bytes from these vnodes and the bytes from 'swap –l', you'll see some difference, I'm working on this to clarify still (I can't explain it yet). Here is a sample output (it runs for a long time, with –d flag you'll see what it does exactly):

```
…
Processing PID 11 (0xb) EXEC :  [ "/lib/svc/bin/svc.configd" ] in zone  0x187a260
"global"
array_chunk db size : 55
an_pvp db size : 77100
/dev/zvol/dsk/rpool/swap       269

Processing PID 9 (0x9) EXEC :  [ "/lib/svc/bin/svc.startd" ] in zone  0x187a260
"global"
array_chunk db size : 66
an_pvp db size : 984
/dev/zvol/dsk/rpool/swap       982

Processing PID 424 (0x1a8) EXEC :  [ "-sh" ] in zone  0x187a260 "global"
array_chunk db size : 10
an_pvp db size : 16
/dev/zvol/dsk/rpool/swap       15

Processing PID 397 (0x18d) EXEC :  [ "/usr/lib/saf/sac -t 300" ] in zone  0x187a260
"global"
array_chunk db size : 14
an_pvp db size : 34
/dev/zvol/dsk/rpool/swap       33

Processing PID 402 (0x192) EXEC :  [ "/usr/lib/saf/ttymon" ] in zone  0x187a260
"global"
array_chunk db size : 22
an_pvp db size : 51
/dev/zvol/dsk/rpool/swap       48

SWAP summary :
/dev/zvol/dsk/rpool/swap       9273
total swap : 9273
```

The swap counters are in pages.

And now **about the experiment**. I wrote a small perl script 'mem_use.pl' which fills up an array with random numbers hence putting memory pressure on the system. Only a few lines and I can control how much memory it should use. So in one window, I have DTrace script running I quoted before (to see page ins from disk swap), a second window I have 'mem_use.pl' running, I will use this to put pressure on memory usage, in a third window I have vmstat 5 running (to see 'sr' column kicks off) and a fourth window where I grep for lotsfree and freemem to see how available memory drops, also running a swap –l query every 5 secs (while true ; do date ; kstat -n system_pages | egrep "lotsfree|freemem" | awk '{printf("%-50s%d\n",$0,($NF*8192)/(1024*1024))}'; swap -l ; sleep 5 ; done).

Phase #1

Baseline.

2nd window (mem_use.pl) :

```
gislabsol04# ./mem_use.pl
```

3rd window (vmstat) :

```
gislabsol04# vmstat 10
kthr      memory            page            disk          faults      cpu
r b w   swap  free  re  mf pi po fr de sr s0 s2 -- --   in   sy   cs us sy id
0 0 0 8451608 510984 2   5  0  0  0  0  0  0  0  0  268   59  196  0  0 100
0 0 0 7855064 36000 61 251  3  0  0  0  0  4  4  0  0  480  504 1174  1  1 98
```

4th window (lotsfree,freemem) :

```
Fri Oct 12 04:01:14 EDT 2012
        freemem                         128460            1003
        lotsfree                        3993              31
swapfile                 dev        swaplo  blocks   free
/dev/zvol/dsk/rpool/swap 256,1      16 1    6777200  16550416
```

Everything looks calm, we have about 1G free memory (128460 pages), demand paging will kick off when we'll get below 31Mbyte memory (3993 pages, lotsfree). You may see that we already have some swap blocks in use even though we have memory available. That's because I had a memory pressure before during my tests, and those pages that were evacuated earlier by the pageout process will get brought back into the memory either when the process which they belong to access them, or the process exit. (And that's why we see 10M of disk swap blocks in use on gisatlvirt47 as well, it's not a problem as it's not in actual use)

Phase #2

I start pushingmemory pressure on the system.

2nd window :

```
up 700
increase up by up 700
added 700 entries, current size 716799
up 200
increase up by up 200
added 200 entries, current size 921599
up 100
increase up by up 100
added 100 entries, current size 1023999
```

The up lines are commands I typed in, meaning "increase memory usage". The unit is irrelevant here, as I was watching the freemem on the 4th window that's why I had to increase usage in multiple steps. I wanted to push usage where the system becomes freemem < lotsfree.

4th window :

```
Fri Oct 12 04:01:24 EDT 2012
        freemem                         82137             641
        lotsfree                        3993              31
swapfile             dev  swaplo blocks   free
/dev/zvol/dsk/rpool/swap 256,1      16 16777200 16550416
```

```
Fri Oct 12 04:01:29 EDT 2012
        freemem                         33688             263
        lotsfree                        3993              31
swapfile             dev  swaplo blocks   free
…
Fri Oct 12 04:01:40 EDT 2012
        freemem                         33693             263
        lotsfree                        3993              31
swapfile             dev  swaplo blocks   free
…
Fri Oct 12 04:02:26 EDT 2012
        freemem                         5987              46
        lotsfree                        3993              31
swapfile             dev  swaplo blocks   free
/dev/zvol/dsk/rpool/swap 256,1      16 16777200 16550480
Fri Oct 12 04:02:31 EDT 2012
       freemem                        2524              19
        lotsfree                        3993              31
swapfile             dev  swaplo blocks   free
/dev/zvol/dsk/rpool/swap 256,1      16 16777200 16404608

Fri Oct 12 04:02:37 EDT 2012
       freemem                        4006              31
        lotsfree                        3993              31
swapfile             dev  swaplo blocks   free
```

As you see I pushed the memory usage so freemem became < lotsfree, and after that happened for some reason we gained more memory (the pageout kicked in…), freemem 263, 46, 19 (less than lotsfree), 31.

3rd window :

```
0 0 0 8826568 973256 56 3082 0 0  0  0  0  2  2  0  0  519 10378 231 14  5 81
0 0 0 8202088 351528 38 4107 0 0  0  0  0  2  2  0  0  528 9314  225 13  6 81
0 0 0 8117720 267488 41 1984 2 0  0  0  0  3  3  0  0  500 3998  219  5  3 92
0 0 0 7915776 63320 53 1115 0  0  0  0  0  2  2  0  0  490 2249  218  3  2 95
0 0 0 7909224 56872 53 239  1  0  0  0  0  6  6  0  0  493  500  240  1  1 98
0 0 0 7909032 56608 53 241  0  0  0  0  0  3  2  0  0  483  534  235  1  1 98
0 0 0 7909224 56808 52 239  0  0  0  0  0  2  2  0  0  488  524  229  1  1 98
1 0 0 7833512 26992 42 1622 31 8050 8991 0 58739 94 95 0 0 817 3286 8806 5 31 64
```

As you see the sr column is increased, indicating that the pageout process started to scan the least recently used pages and evacuate them from the physical memory to physical disk. (That's how we get from 19M freemem to 31M freemem, even though we did not quit any process!)

Phase #3

I quit the 'mem_use.pl' tool, so the system is relieved from the memory pressure and you'll see pages are brought back into memory from disk swap as they are accessed by the processes:

```
2012 Oct 12 04:04:01 exec: zsh pid: 7987 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:01 exec: zsh pid: 7987 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:01 exec: zsh pid: 7987 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:01 exec: zsh pid: 7987 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:01 exec: utmpd pid: 420 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:01 exec: utmpd pid: 420 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:01 exec: utmpd pid: 420 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:04 exec: nscd pid: 145 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:05 exec: zsh pid: 7975 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:05 exec: zsh pid: 7975 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
```

```
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: nscd pid: 145 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
2012 Oct 12 04:04:06 exec: svc.configd pid: 11 uid: 0 size: 8192 maj : 256 min : 1
```

etc.

Summary :

That's all I wanted to show you. The **goal of the experiment was** to show exactly at which cases will Solaris use disk swap (demand paging and swap out), how can you monitor it (sr column in vmstat is the best way), also if you see some used blocks in 'swap –l' doesn't mean anything bad if you don't see the sr column with number higher than 0, it only means sometimes in the past you had demand paging kicked in. Page in from disk swap occurs only when the process tries to access that memory area or the process quits. Also the tool I wrote shows me that whenever the paging occurs, the pageout process from a process standpoint really randomly picks pages using the LRU algorithm to move them to physical disk, so there is a good change that whenever you see a number in the sr column, the chances are good that most of the processes are having some pages moved out to the physical disk. One final note about the pageout process, even though it does not understand processes, it has a built in feature that the pages which are shared will less likely to be paged out, because those are most likely libraries used by more processes. The way Solaris handles this is that every page has a reference counter on how many processes are using that page. Other than that, the pageout process deals with pages only, it has no information whatsoever about who they belong to.

Regards,
sendai

From: Field, Nathen (GE Capital)
Sent: Wednesday, October 03, 2012 3:04 PM
To: Spitzer, Andras (GE, Corporate)
Subject: RE: VM SWAP

Thanks, do you have any one-liners to find who is consuming swap ?

From: Spitzer, Andras (GE, Corporate)
Sent: Tuesday, October 02, 2012 11:38 PM
To: Field, Nathen (GE Capital)
Subject: RE: VM SWAP

Nate,

Let me explain. First of all, arcstat.pl makes not much sense in a zone environment, as it reads the arc cache counters from the kernel (using kstat), and as zones have no individual kernels you'll see system wide stats, not that are specific to your own zone. Second, I see you run top. Please don't, top was never meant to understand Solaris so it will just confuse you. Run prstat, and the other native stat tools within Solaris. The confusion here is around swap. Swap is understood very differently by UNIXes, and what top shows you even though it's true it's not what you think.

First of all, let's see your zone's configuration :

```
gisatlvirt47# /usr/sbin/zonecfg -z cmfalgadwps05v
zonecfg:cmfalgadwps05v> info
…
capped-memory:
        physical: 32G
```

```
        [swap: 64G]
```

This means your zone has 32G physical memory, and 64G swap. Here comes the tricky part. What swap means? Swap in Solaris is equal to “physical memory space” + “swap disk space”. This is the tricky part, as traditionally swap used to mean physical disk backed virtual memory, but in Solaris this is different. In Solaris swap is equal the virtual memory Solaris has, which is equal to total physical memory available + total disk swap available, or as Solaris calls it “physical memory swap + disk space swap”. Also, the swap size will set the /tmp as well.

So there are two ways to query the swap in Solaris, first you can query the total swap with ‘swap –s’ :

```
cmfalgadwps05v% /usr/sbin/swap -s
total: 29216752k bytes allocated + 0k reserved = 29216752k used, 37892112k available
```

and the other way is to show only the disk based swap with ‘swap –l’ :

```
cmfalgadwps05v% /usr/sbin/swap -l
swapfile             dev  swaplo blocks   free
/dev/swap            0,0      16 134217728 75784224
```

(please note as swap –l counts in disk blocks, one block equal to 512 bytes). Now this differentiation works in global zone well, not in local zones anymore. The reason is that in zones as you really don’t have disk swap device configured, by default you only inherit the disk swap from the global zone and ‘swap –l’ will show you the size of a swap pseudo device, not a real disk based swap device as it would in global zone. Yea, it’s confusing a bit ☺.

Also, a rule of thumb in Solaris is that it will use physical memory always first, it only will use disk based swap if the OS is running out of physical memory. So how can we understand the configuration of 32G physical memory and 64G swap?

Well, your zone will use first and foremost the 32G physical memory, and if it runs out of physical memory, it can still use disk based swap space for additional 32G. The calculation is simple, 64G swap which is equal to physical memory + disk swap, and we just set the physical memory to 32G, it leaves us with 32G disk swap. in other words, we have set the physical memory size and the swap size, and you can get the disk swap as = (swap – physical), which is 32G. And your zone will only start to use the 32G disk swap as it runs out of 32G physical memory.

Hope it makes sense ☺

Btw, I can see your zone using 28G physical memory from the global zone :

```
gisatlvirt47% rcapstat -z 5 | egrep "zone|05"
    id zone             nproc    vm   rss   cap    at avgat    pg avgpg
     3 cmfalgadwps05v       -   28G   21G   32G    0K    0K    0K    0K
```

You see it’s cap at 32G. And your 28G used memory is equal to the virtual memory used :

```
cmfalgadwps05v% /usr/sbin/swap -s
total: 29216880k bytes allocated + 0k reserved = 29216880k used, 37891984k available
```

If you add up used (29216880k) and available swap space (37891984k), you’ll get the total size of the virtual memory (67108864k), which is the swap we set earlier to 64G. Finally, let’s see the confusing ‘swap –l’ which should show us the block devices, but as we are in a zone it will show a pseudo swap device :

```
cmfalgadwps05v% /usr/sbin/swap -l
swapfile             dev  swaplo blocks   free
/dev/swap            0,0      16 134217728 75783968
```

Now, normally this would show us the disk swap, but as we are in a zone and it shows us a pseudo swap device, I expect this to be equal to the total swap, or in other words equal to ‘swap –s’. Let’s calculate :

cmfalgadwps05v% bc

134217728-75783968 (number of blocks – free blocks)
58433760 (will give us the used blocks)
58433760*512 (multiplied by 512 as these are blocks)
29918085120 (which is ~28G memory, which is equal to what we saw before)

so 'swap –l' is not very useful in a local zone as it will show a pseudo swap device, not a real disk device as it would in a global zone.

Also, to triple-confirm your zone does not use disk based swap, you remember your zone does not have swap device, it inherits disk swap from the global zone, let me show you the disk swap usage of the global zone :

```
gisatlvirt47% /usr/sbin/swap -l
swapfile             dev  swaplo blocks   free
/dev/zvol/dsk/rpool/swap 256,1      16 16777200 16768400
/dev/zvol/dsk/rpool/swap1 256,3      16 524287984 524276528
```

You see we are using two swap devices. Now let's do the math how much disk swap blocks are in use :

gisatlvirt47% bc
16777200-16768400 (total blocks – free blocks will give us the used blocks)
8800 (used blocks for swap)
524287984-524276528 (same for swap1)
11456 (used blocks for swap1)
8800+11456 (we sum up swap and swap1 used block counts)
20256 (result ☺)
20256*512 (is in blocks so let's get the amount in bytes)
10371072 (the result in bytes)
10371072/(1024*1024) (the result in Mbytes)
9

So you can see our global zone is using 9MByte disk swap, this is used by another local zone who is using up all their physical memory, still it's almost nothing.

Nate, in short your zone is working fine, it does not use disk swap at all, it is using physical memory at this point, although it's running on ~90% memory utilization (28G/32G), and it still has 32G disk swap ready to kick in if you run out of physical memory, although that would impact performance badly, you know that. Please let me know if I can help further.

Regards,
sendai