Subject: Using threads on OpenVZ, and memory allocation versus memory usage
Posted by MvdS on Sun, 06 May 2007 14:07:39 GMT
View Forum Message <> Reply to Message

I noticed NPTL allocates huge amounts of memory for each thread. 8MiB per thread is not uncommon. Of course with a privvmpages limit set on a VE, starting a program with threads can easily consume all privvmpages.

Setting privvmpages is a common (and advised) way to limit the amount of RAM a VE is able to use. But it is both unrelated to the RAM usage and unfair against applications allocating without using the allocated memory, NPTL is a widly used example of this. Furthermore, with a vanilla linux kernel, the allocation is only limited by the address space op the application. This means, on a 32bit system, the typical limit is 3GiB/process. On a typical OpenVZ system this is only privvmpages for the complete VE.

Portable code should not count on the behavior specific to the kernel, however NPTL is made in collaboration with the linux kernel, and does expect this implementation detail.

OpenVZ breaks this behavior, and thereby greatly reduces the usability of threads within a VE.

I've been discussing this behavior with an OpenVZ system administrator, and there seems no sane way to limit the amount of available RAM.
It is possible to use oomguarpages to guarantee some amount of RAM+swap, but it seems not possible to limit it.

With only oomguarpages configured as a limit for system resources the less than ideal situation that one VE uses all RAM+swap, while other VE's are below their limits, is possible. Because of the nature of swapping, this slows down the complete system. Also, the less active VE's might be totally swapped-out, which means reduced responsiveness, because of 1 misbehaving VE.

This might be resolved by implementing a hard limit on the RAM+swap counter, relating to oomguarpages in the same way as privvmpages relates to vmguarpages. The future releases that might support a settable physpages might help, but is not quite the same.

My direct questions about this matter are:
1. Are there any workarounds to emulate the vanilla kernel behavior without losing control of VE's memory usage?
1. If not, are the OpenVZ developers aware of the 'punishment' users of threaded applications get?
2. What is the opinion across the OpenVZ user base about this?
3. Is there any development in progress to remedy this situation?

PS: Here is a sample application to reproduce the behavior:
```
/* to compile:
 * $ LDFLAGS=-lpthread make dummy-threads
 */
#include <stdio.h>
#include <stdlib.h>
```

```
#include <pthread.h>
#define NUM_THREADS 131072

void *thread_routine(void *data){}

int main(int argc, char **argv){
  pthread_t thread;
  int i;
  int r = 0;

  printf("Trying to create %i threads\n", NUM_THREADS);
  for(i=0;i<NUM_THREADS && !r;i++){
    r = pthread_create(&thread, NULL, thread_routine, NULL);
    if(r) perror("pthread_create");
  }
  printf("A total of %i threads created\n", i);
  pause();
}
```

On 32bit linux this typicaly creates 300+ threads, on 64bit linux this in 3000+, but with OpenVZ this depends on privvmpages and other applications running within the VE.
Also, running the program 2 times on linux does not matter, on OpenVZ the second program is only able to start 1 thread.

Subject: Re: Using threads on OpenVZ, and memory allocation versus memory usage
Posted by xemul on Mon, 07 May 2007 08:44:47 GMT
View Forum Message <> Reply to Message

Nowadays we do not have any possibility to set the limit on physical memory usage. But we are working on per-VE RSS limiting, i.e. you set a limit, and when VE hits one (with physical memory usage, not with privvmpages), the kernel starts to swap-out or shrink pages on disk from this VE's only.

We have a beta version of this patch posted on LKML, but it is not yet accepted and commented by kernel maintainers.

My question is - what memory is allocated? I would appreciate if you show me the differences of /proc/<pid>/mmap when new threads appear. With this knowledge I could answer your questions with more detail.

Thank you.

Subject: Re: Using threads on OpenVZ, and memory allocation versus memory usage

The above is a proof of concept program, but may be you like something from real live? On the internet you can find many people who have trouble installing Apache 2 mpm worker on Virtuozzo/OpenVZ based VE's with a privvmpages of around 512 MiB. The same applied on my virtual servers which are running on OpenVZ. So I investigated and found the following:

[1] very minimalistic server, only 4 processes (init, cron, syslogd and OpenSSH)

```
privvmpages        663       972    131072    147456         0
physpages          486       739         0   (2^63-1)         0
```

As you can see, hardly any memory is being used, and physpages is almost the same as privvmpages. Now I install Apache 2 mpm worker (Debian package). On a normal server which isn't stripped down like mine, you run into problems. On this stripped down server and 512 MiB burstable memory it is just possible. After a reboot, I see the following:

```
privvmpages     112933     122415     131072     147456         0
physpages         1573      12562          0   (2^63-1)         0
```

As you can see, there is a huge difference between physpages and privvmpages. You can't do anything with this server, because it is almost out om memory. Starting aptitude for example crashes. When I do a pmap on the four instances (two are control forks, two start in the deafult configuration 25 threads each) I see the following:

```
ve35:/# pmap 26120
26120:   /usr/sbin/apache2 -k start -DSSL
...
 total     7224K
ve35:/# pmap 26121
26121:   /usr/sbin/apache2 -k start -DSSL
...
 total     7016K
ve35:/# pmap 26122
26122:   /usr/sbin/apache2 -k start -DSSL
...
405dd000   8188K rwx--    [ anon ]
40ddc000      4K -----    [ anon ]
  => 27 times
...
 total   228444K
ve35:/# pmap 26127
26127:   /usr/sbin/apache2 -k start -DSSL
```

...
405dd000   8188K rwx--   [anon]
40ddc000   4K -----   [anon]
  => 27 times
...
 total   228444K

So every thread uses a buffer of 8 MiB. Most of the times it doesn't use thsi. On a normal machine with 512 MiB available RAM this is not a problem. You can start aptitude and a lot more programs. The only solutions is configuring Apache to use less default waiting threads. In my opinion 50 is way to much for most servers, but this is the default. And according to physpages and a free on the hardware node, it is not a real problem that this huge amount disappears in buffers...

This strange behavior between a server with real RAM and a OpenVZ server with the same amount is seen in more situations. I even asked about this in this forum but got no answer. I now know what is the problem and most of the times I can work around it. But is always a discussion with clients, because they se it working on a normal server.

Subject: Re: Using threads on OpenVZ, and memory allocation versus memory usage
Posted by xemul on Thu, 10 May 2007 07:53:18 GMT
View Forum Message <> Reply to Message

> As you can see, there is a huge difference between physpages and privvmpages.

I do see this. This means that the program mmap-s too many memory and doesn't use it at all. My question was to show what kind of mappings eat so many of privvmpages. Maybe this issue is somewhat specific and can be treated specially in the kernel. This may allow such application to run.

Subject: Re: Using threads on OpenVZ, and memory allocation versus memory usage
Posted by hvdkamer on Thu, 10 May 2007 08:19:18 GMT
View Forum Message <> Reply to Message

xemul wrote on Thu, 10 May 2007 09:53My question was to show what kind of mappings eat so many of privvmpages.

I'm not an developer. But that is why I gave the example of Apache 2 mpm worker. Probably someone with knowledge could find out were the 54 mappings in the source are made? It is probably an small prortion of the code?

However Apache is not the only program. On my VE's clients use lots of different programs and I see in all a discrepancy between the two rangingfor 1:4 to 1:80. At this moment I'm thinking that I

can probably raise the privvmpages on the assumption that privvmpages is for at least a large part not used. But as systemadministrator I don't like guesses, so a better way to handle this effect would be very helpfull...

Subject: Re: Using threads on OpenVZ, and memory allocation versus memory usage
Posted by xemul on Thu, 10 May 2007 09:11:13 GMT
View Forum Message <> Reply to Message

Quote:I'm not an developer. But that is why I gave the example of Apache 2 mpm worker. Probably someone with knowledge could find out were the 54 mappings in the source are made? It is probably an small prortion of the code?

I have pointed what knowledge will help in finding this out - the sequential dumps of /proc/PID/maps file for multi threaded application.

The same in case of Apache - just show us the contents of this file.

Thanks.

Subject: Re: Using threads on OpenVZ, and memory allocation versus memory usage
Posted by hvdkamer on Thu, 10 May 2007 09:44:39 GMT
View Forum Message <> Reply to Message

xemul wrote on Thu, 10 May 2007 11:11the sequential dumps of /proc/PID/maps file for multi threaded application.

I would love to help, but when Apache starts I have no idea how to get the sequential dumps. I can give you the dumps after it has started.

Subject: Re: Using threads on OpenVZ, and memory allocation versus memory usage
Posted by MvdS on Tue, 15 May 2007 07:41:26 GMT
View Forum Message <> Reply to Message

I have had no time yet to show the changes in the memmory map on an OpenVZ machine. But I found one thing that helps, limiting the stack size with ulimit:
ulimit -s 16
Significantly reduces the stack used by every thread, and thus the amount of memory allocated. However I have no idea what is a good amount of stack to use. I suppose this differs with programs.

The quick fix (while waiting for the RSS limiting) would be to mention the usage of threads and ulimit in the documentation of privvmpages usage.

pthreads allows to set the stack size, but it defaults to the maximum allowed by ulimit.

The mappings are really straight forward (4K + around 8MiB per thread). If you still want this info, I'll probably be able to mesure it sometime this week.

Subject: Re: Using threads on OpenVZ, and memory allocation versus memory usage
Posted by xemul on Mon, 21 May 2007 13:25:53 GMT
View Forum Message <> Reply to Message

These 8MiBs per thread are the most interesting ones. Stack is stack, but this extra memory looks larger...
