Subject: Linux Memory Overcommit in OpenVZ
Posted by charlesl on Thu, 19 Jun 2008 13:00:37 GMT
View Forum Message <> Reply to Message

After thinking I'd fully understood the three main UBC memory parameters vmguarpages, privvmpages and oomguarpages, I find myself asking another question as follows (note for anyone unfamiliar with the standard Linux kernel overcommit, this is a good explanation: Linux Memory Overcommit).

So we have the following barriers/limits:
vmguarpages - The guaranteed memory for the VE (when allocating memory)
privvmpages - Sets an upper bound on the memory for the VE (when allocating memory)
oomguarpages - Usually the same as vmguarpages, sets the guaranteed limit under OOM condition (when de-allocating memory).
Further, these are the accounting values:
vmguarpages - unused
privvmpages - The total number of pages allocated + used
oomguarpages - The total number of pages used in RAM + swap
physpages - The total number of pages used in RAM

Okay, so that's how I understand things to work (please tell me if I'm wrong!).

Now, my question is about the default Linux memory overcommit - in an unmodified kernel, this allows an application to make any mallocs it likes successfully, then (in some OS dependent manner) deny applications memory when they come to actually use it if there are insufficient resources. BUT, OpenVZ counts allocations in its privvmpages held value so places an upper bound on the amount which may be allocated (even if that is not being used).

So these seem to be conflicting: on the one hand the Linux kernel doesn't care about allocations (only about used memory) and allows virtually all mallocs to succeed regardless of the memory situation, while on the other OpenVZ seems to account for (and prevent?) allocations above the barrier of privvmpages.

I also notice the OpenVZ wiki on UBC says that the total of all privvmpages barriers can be above the total RAM + Swap of the system for the very reason that the kernel doesn't care about mallocs, only about used memory. Does this mean a VE can safely allocate over privvmpages barrier (indeed over RAM + swap capacity of the machine) provided it is not all used, or will OpenVZ not allow this?

So which of these is true: is the OpenVZ kernel modified so that it really does take account of allocations too, or is it true that in an OpenVZ VE all mallocs will always succeed and it doesn't limit them?

Thanks in advance for any thoughts/explanations.

Subject: Re: Linux Memory Overcommit in OpenVZ

Posted by charlesl on Sat, 12 Jul 2008 13:59:50 GMT
View Forum Message <> Reply to Message

It has been nearly a month, almost 1000 views and not a single reply so I guess this has everyone else stumped too (or its so obvious it didn't deserve an answer ) ...

I wrote a couple of programs to test and I'll share my results. The two programs were:

memalloc - Allocates memory in 100MB/second increments
memfill - Allocates and fills memory at rate 100MB/second
I then ran these in a VE and watched its user_beancounters from the HN, and the HN's /proc/meminfo too. Here are my findings:

memalloc
The privvmpages accounting increases until it hits the privvmpages barrier, at which failcnt increases and the program begins to fail.

But /proc/meminfo does not change - the kernel is not counting any allocations made via malloc.

memfill
The privvmpages accounting and physpages/oomguarpages accounting all increase. Since this is a test system not under load, the privvmpages barrier is reached first, its failcnt increases and the program fails.

In this case, /proc/meminfo does change - the kernel is counting used pages.

Conclusion

Note: naturally, "allocated" memory includes "used" memory. An allocation limit is implicitly a usable limit too. I use _bar for barrier and _cur for current (held) values.

The VE has a maximum allocation limit of privvmpages_bar.
The VE has a guaranteed allocation limit of guarvmpages_bar.
The kernel only cares about used memory - the sum of all VE's physpages_cur/oomguarpages_cur (+ HN overhead).
VEs can allocate all the memory they like up to privvmpages_bar under situations where the total used memory is low.
VEs using memory above their guarantees may fail above the guarantees under high overall memory consumption.

My only one remaining question is: clearly VEs cannot use arbitrary amounts of memory under high load conditions, but can VEs still allocate up to their privvmpages_bar even under high memory conditions, since the kernel doesn't care about allocations?

Subject: Re: Linux Memory Overcommit in OpenVZ

Posted by charlesl on Sat, 12 Jul 2008 20:27:14 GMT
View Forum Message <> Reply to Message

After further tests (using the previous programs I documented above under HN low memory availability), the answer to my earlier question is: yes. I can summarise my conclusions as follows ("memory" = RAM + Swap here):
A VE may allocate any amount it likes up to privvmpages_bar, since the kernel doesn't account this memory and therefore there is no effect on the HN.
If a VE tries to use more memory in an HN-low-memory situation, and is over its guarantee of vmguarpages_bar, its request for allocation is rejected.
If a VE tries to allocate + use memory in an HN-low-memory situation, but is below its vmguarpages_bar, the request will succeed and processes in other containers will be killed to compensate under OOM conditions.
If the previous happens, the HN is out of memory (OOM) and OpenVZ must trim back on containers using more than their guaranteed barriers. It will therefore reduce processes in containers down to their oomguarpages_bar until there is sufficiently free memory (starting with the container in largest excess). Quite how it decides which processes to kill I don't know, nor particularly care (though feel free to add this information!) However, it will kill all processes it sees fit which may include any which purely allocate memory and do not use it (as happened in my tests). Although killing processes made purely of allocated memory seems pointless (since it makes no difference to the free memory in the kernel), in reality all programs will use around 50% their total allocated memory - thus I guess it isn't necessary to discriminate which processes to kill based on the ratio of used/allocated memory on a process-by-process basis. Though this could be an enhancement to the OpenVZ OOM algorithm I suppose - attempt to kill the fewest processes possible, so start by killing those using (rather than allocating) the largest amount of memory.
After an OOM has occurred, VEs may continue to allocate any amount of memory up to vmguarpages_bar without problem - if another OOM occurs (as a result of using this memory) these processes may again be killed.
This is all especially important when using programs that allocate far more than they ever use (and there are a lot of those) - but as long as the server never hits OOM it's all safe and good.
That's all perhaps a bit detailed for most people, but I like to feel I know the system I'm using inside out

Subject: Re: Linux Memory Overcommit in OpenVZ
Posted by kir on Thu, 17 Jul 2008 07:42:32 GMT
View Forum Message <> Reply to Message

Hi,

It would be great if you post your findings on wiki