Subject: Containers and /proc/sys/vm/drop_caches
Posted by Mike Hommey on Wed, 05 Jan 2011 09:40:22 GMT
View Forum Message <> Reply to Message

[Copy/pasted from a previous message to lkml, where it was suggested to
try containers@]

Hi,

I noticed that from within a lxc container, writing "3" to
/proc/sys/vm/drop_caches would flush the host page cache. That sounds a
little dangerous for VPS offerings that would be based on lxc, as in one
VPS instance root user could impact the overall performance of the host.
I don't know about other containers but I've been told openvz isn't
subject to this problem.
I only tested the current Debian Squeeze kernel, which is based on
2.6.32.27.

Cheers,

Mike

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

Subject: Re: Containers and /proc/sys/vm/drop_caches
Posted by Rob Landley on Fri, 07 Jan 2011 13:03:48 GMT
View Forum Message <> Reply to Message

On 01/06/2011 03:43 PM, Matt Helsley wrote:
> On Wed, Jan 05, 2011 at 07:46:17PM +0530, Balbir Singh wrote:
>> On Wed, Jan 5, 2011 at 7:31 PM, Serge Hallyn <serge.hallyn@canonical.com> wrote:
>>> Quoting Daniel Lezcano (daniel.lezcano@free.fr):
>>>> On 01/05/2011 10:40 AM, Mike Hommey wrote:
>>>>> [Copy/pasted from a previous message to lkml, where it was suggested to
>>>>> try containers@]
>>>>>
>>>>> Hi,
>>>>>
>>>>> I noticed that from within a lxc container, writing "3" to
>>>>> /proc/sys/vm/drop_caches would flush the host page cache. That sounds a
>>>>> little dangerous for VPS offerings that would be based on lxc, as in one
>>>>> VPS instance root user could impact the overall performance of the host.
>>>>> I don't know about other containers but I've been told openvz isn't
>>>>> subject to this problem.
>>>>> I only tested the current Debian Squeeze kernel, which is based on

>>>>> 2.6.32.27.
>>>>
>>>> There is definitively a big work to do with /proc.
>>>>
>>>> Some files should be not accessible (/proc/sys/vm/drop_caches,
>>>> /proc/sys/kernel/sysrq, ...) and some other should be virtualized
>>>> (/proc/meminfo, /proc/cpuinfo, ...).
>>>>
>>>> Serge suggested to create something similar to the cgroup device
>>>> whitelist but for /proc, maybe it is a good approach for denying
>>>> access a specific proc's file.
>>>
>>> Long-term, user namespaces should fix this - /proc will be owned
>>> by the user namespace which mounted it, but we can tell proc to
>>> always have some files (like drop_caches) be owned by init_user_ns.

Changing ownership so a script can't open a file that it otherwise
could may cause scripts to fail when run in a container. Makes the
containers less transparent.

>>> I'm hoping to push my final targeted capabilities prototype in the
>>> next few weeks, and after that I start seriously attacking VFS
>>> interaction.
>>>
>>> In the meantime, though, you can use SELinux/Smack, or a custom
>>> cgroup file does sound useful. Can cgroups be modules nowadays?
>>> (I can't keep up) If so, an out of tree proc-cgroup module seems
>>> like a good interim solution.
>>>
>>
>> Ideally a drop_cache should drop page cache in that container, but
>> given container have a lot of shared page cache, what is suggested
>> might be a good way to work around the problem
>
> One gross hack that comes to mind: Instead of a hard permission model
> limit the frequency with which the container could actually drop caches.
> Then the container's ability to interfere with host performance is more
> limited (but still non-zero). Or limit frequency on a per-user basis
> (more like Serge's design) because running more containers by a
> compromised user account shouldn't allow more frequent cache dropping.

Disk access causes at best multi-milisecond latency spikes, which can cause
a heavily loaded server to go into thrashing meltdown. So a container
could screw up another container with this pretty badly.

The easy short-term fix is to make containers silently ignore writes to
drop_caches.

> That said, the more important question is why should we provide
> drop_caches inside a container? My understanding is it's largely a
> workload-debugging tool and not something meant to truly solve
> problems.

A heavily loaded system that goes deep into swap without triggering the OOM killer can become pretty useless. My home laptop with 2 gigs of ram gets so sluggish whenever I compile something that you can't use the touchpad anymore because hitting the boundary of a widget with the mouse pointer causes a 5 second freeze while it bounces a off three or four processes to handle the message, evicting yet more pages to fault in the pages to handle the X events. By the time the pointer moves again it's way overshot. (Ok, having firefox, chrome, and kmail open with several dozen tabs open in each may have something to do with this.)

When it does this, ctrl-alt-f1 echo 1 > /proc/sys/vm/drop_caches is just about the only thing that will snap it out of it short of killing processes. The system has ~600 megs of ram tied up in disk cache while being so short of anonymous pages the mouse is useless.

That doesn't necessarily apply to containers but that's one use case of using it as a stick to hit the darn overburdened machine when it's making stupid memory allocation decisions. (Playing with swappiness puts the OOM killer on a hair trigger, depending on kernel version du jour.)

However, it's not guaranteed to do anything (the cached data could be dirty, mmaped by some process, immediately faulted back in by some other process), so ignoring writes to drop_caches from a container is probably legal behavior anyway.

Rob

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: Containers and /proc/sys/vm/drop_caches
Posted by Rob Landley on Sat, 08 Jan 2011 12:39:31 GMT
View Forum Message <> Reply to Message

On 01/07/2011 09:12 AM, Serge Hallyn wrote:
>> Changing ownership so a script can't open a file that it otherwise
>> could may cause scripts to fail when run in a container. Makes
>> the containers less transparent.

>
> While my goal next week is to make containers more transparent, the
> official stance from kernel summit a few years ago was: transparent
> containers are not a valid goal (as seen from kernel).

Do you have a reference for that? I'm still coming up to speed on all this. Trying to collect documentation...

>> A heavily loaded system that goes deep into swap without triggering
>> the OOM killer can become pretty useless. My home laptop with 2
>> gigs
>
> Isn't a cgroup that controls both memory and swap access the right
> answer to this?

There are other ways to work around it, sure. (It's yet to be proven that they do actually work better in resource constrained desktop environments under real-world load, but they seem very promising.)

I was just pointing out that this has seen some use as a recovery mechanism, slightly less drastic than the OOM killer. (Didn't say it was a _good_ use. Also, error avoidance and error recovery are different issues, and virtual memory is an inherently overcommitted resource domain.)

> (And do we have that now, btw?)

I think it's coming, rather than actually here. (I thought the beancounters stuff was OpenVZ, controlled by syscalls that the kernel developers rejected. Have resource constraints on anything other than scheduler made it into vanilla yet? If so, what's the UI to control them?)

By the way, from a UI perspective, most of the containers stuff I've seen so far is apparently aimed at big iron deployments (or attempts to make PC clusters look like mainframes, I.E. this "cloud" stuff). I'm glad to see more diverse uses of it, but one of the downsides of cobbling together a mechanism from a dozen different unrelated pieces of infrastructure (clone flags, cgroup filesystem, extra mount flags on proc and such so they behave differently) is that we need a lot of documentation/example code/libraries to make it easy to use. "You can do X" and "it's easy to reliably do X" have a gap that may take a while to close...

Rob