Subject: Re: controlling mmap()'d vs read/write() pages
Posted by ebiederm on Fri, 23 Mar 2007 10:12:09 GMT
View Forum Message <> Reply to Message

Nick Piggin <nickpiggin@yahoo.com.au> writes:

> Eric W. Biederman wrote:
>> Dave Hansen <hansendc@us.ibm.com> writes:
>>
>>
>>>So, I think we have a difference of opinion. I think it's _all_ about
>>>memory pressure, and you think it is _not_ about accounting for memory
>>>pressure. :) Perhaps we mean different things, but we appear to
>>>disagree greatly on the surface.
>>
>>
>> I think it is about preventing a badly behaved container from having a
>> significant effect on the rest of the system, and in particular other
>> containers on the system.
>
> That's Dave's point, I believe. Limiting mapped memory may be
> mostly OK for well behaved applications, but it doesn't do anything
> to stop bad ones from effectively DoSing the system or ruining any
> guarantees you might proclaim (not that hard guarantees are always
> possible without using virtualisation anyway).
>
> This is why I'm surprised at efforts that go to such great lengths
> to get accounting "just right" (but only for mmaped memory). You
> may as well not even bother, IMO.
>
> Give me an RSS limit big enough to run a couple of system calls and
> a loop...

Would any of them work on a system on which every filesystem was on
ramfs, and there was no swap? If not then they are not memory attacks
but I/O attacks.

I completely concede that you can DOS the system with I/O if that is
not limited as well.

My point is that is not a memory problem but a disk I/O problem which is
much easier to and cheaper to solve. Disk I/O is fundamentally a slow
path which makes it hard to modify it in a way that negatively affects
system performance.

I don't think with a memory RSS limit you can DOS the system in a way
that is purely about memory. You have to pick a different kind of DOS
attack.

As for virtualization that is what a kernel is about virtualizing it's resources so you can have multiple users accessing them at the same time. You don't need some hypervisor or virtual machine to give you that. That is where we start. However it was found long ago that global optimizations give better system through put then the rigid systems you can get with hypervisors. Although things are not quite as deterministic when you optimize globally. They should be sufficiently deterministic you can avoid the worst of the DOS attacks.

The real practical problem with the current system is that nearly all of our limits are per process and applications now span more than one process so the limits provided by linux are generally useless to limit real world applications. This isn't generally a problem until we start trying to run multiple applications on the same system because the hardware is so powerful. Which the namespace work which will allow you to run several different instances of user space simultaneously is likely to allow.

At the moment I very much in a position of doing review not implementing this part of it. I'm trying to get the people doing the implementation to make certain they have actually been paying attention to how their proposed limits will interact with the rest of the system. So far generally the conversation has centered on memory limits because it seems that is where people have decided the conversation should focus. What I haven't seen is people with the limitations coming back to me tearing my arguments apart and showing or telling me where I'm confused. In general I can challenge even the simplest things and not get a good response. All of which tells me the implementations are not ready.

I do have some practical use cases and I have some clue how these subsystems work, and I do care. Which puts in a decent position to at least to high level design review.

My biggest disappointment is that none of this is new, and that we seem to have forgotten a lot of the lessons of the past.

Eric

_____