Subject: Re: [RFC] [-mm PATCH] Memory controller fix swap charging context in unuse_pte()
Posted by Hugh Dickins on Thu, 25 Oct 2007 19:33:36 GMT
View Forum Message <> Reply to Message

On Wed, 24 Oct 2007, Balbir Singh wrote:
> Hugh Dickins wrote:
> >
> > Thanks, Balbir. Sorry for the delay. I've not forgotten our
> > agreement that I should be splitting it into before-and-after
> > mem cgroup patches. But it's low priority for me until we're
> > genuinely assigning to a cgroup there. Hope to get back to
> > looking into that tomorrow, but no promises.
>
> No Problem. We have some time with this one.

Phew - I still haven't got there.

> > I think you still see no problem, where I claim that simply
> > omitting the mem charge mods from mm/swap_state.c leads to OOMs?
> > Maybe our difference is because my memhog in the cgroup is using
> > more memory than RAM, not just more memory than allowed to the
> > cgroup. I suspect that arrives at a state (when the swapcache
> > pages are not charged) where it cannot locate the pages it needs
> > to reclaim to stay within its limit.
>
> Yes, in my case there I use memory less than RAM and more than that
> is allowed by the cgroup. It's quite possible that in your case the
> swapcache has grown significantly without any limit/control on it.
> The memhog program is using memory at a rate much higher than the
> rate of reclaim. Could you share your memhog program, please?

Gosh, it's nothing special. Appended below, but please don't shame
me by taking it too seriously. Defaults to working on a 600M mmap
because I'm in the habit of booting mem=512M. You probably have
something better yourself that you'd rather use.

> In the use case you've mentioned/tested, having these mods to
> control swapcache is actually useful, right?

No idea what you mean by "these mods to control swapcache"?

With your mem_cgroup mods in mm/swap_state.c, swapoff assigns
the pages read in from swap to whoever's running swapoff and your
unuse_pte mem_cgroup_charge never does anything useful: swap pages
should get assigned to the appropriate cgroups at that point.

Without your mem_cgroup mods in mm/swap_state.c, unuse_pte makes

the right assignments (I believe). But I find that swapout (using
600M in a 512M machine) from a 200M cgroup quickly OOMs, whereas
it behaves correctly with your mm/swap_state.c.

Thought little yet about what happens to shmem swapped pages,
and swap readahead pages; but still suspect that they and the
above issue will need a "limbo" cgroup, for pages which are
expected to belong to a not-yet-identified mem cgroup.

>
> Could you share your major objections at this point with the memory
> controller at this point. I hope to be able to look into/resolve them
> as my first priority in my list of items to work on.

The things I've noticed so far, as mentioned before and above.

But it does worry me that I only came here through finding swapoff
broken by that unuse_mm return value, and then found one issue
after another. It feels like the mem cgroup people haven't really
thought through or tested swap at all, and that if I looked further
I'd uncover more.

That's simply FUD, and I apologize if I'm being unfair: but that
is how it feels, and I expect we all know that phase in a project
when solving one problem uncovers three - suggests it's not ready.

Hugh

```
/* swapout.c */
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <sys/mman.h>

int main(int argc, char *argv[])
{
 unsigned long *base = (unsigned long *)0x08400000;
 unsigned long size;
 unsigned long limit;
 unsigned long i;
 char *ptr = NULL;

 size = argv[1]? strtoul(argv[1], &ptr, 0): 600;
 if (size >= 3*1024)
  size = 0;
 size *= 1024*1024;
 limit = size / sizeof(unsigned long);
```

```
 if (size == 0 || base + limit + 1024 > &size) {
  errno = EINVAL;
  perror("swapout");
  exit(1);
 }
 base = mmap(base, size, PROT_READ|PROT_WRITE,
      MAP_ANONYMOUS|MAP_PRIVATE, -1, 0);
 if (base == (unsigned long *)(-1)) {
  perror("mmap");
  exit(1);
 }
 for (i = 0; i < limit; i++)
  base[i] = i;
 if (ptr && *ptr == '.') {
  printf("Type <Return> to continue ");
  fflush(stdout);
  getchar();
 }
 for (i = 0; i < limit; i++)
  base[i] = limit - i;
 return 0;
}
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers