Subject: chkpnt works, resume fails, same hardware - big ram image, ideas?
Posted by minektur on Wed, 07 Apr 2010 17:59:52 GMT
View Forum Message <> Reply to Message

Ok - I have the following setup:

Kernel: 2.6.18-164.11.1.el5.028stab068.5
vzctl: version 3.0.23
template: centos-5-x86_64 (but issue with debian-5.0-x86_64 too)
(filesystem is GFS cluster)

I checkpoint and try to resume on the same machine 5 seconds later. The checkpoint appears to work, the restore fails.

It has taken me a while to narrow down, but I now have a simple test case that causes this error: If I have a single process with larger than about 2G of RSS, I can checkpoint just fine but when I resume, I get errors like:

```
-----------------------------------------------
% vzctl restore 1050051
Restoring container ...
Starting container ...
Container is mounted
      undump...
Adding IP address(es): XXX.XXX.XXX.XXX
...
Setting CPU limit: 400
Setting CPU units: 6400
Configure meminfo: 2097152
Error: undump failed: Bad file descriptor
Restoring failed:
Error: do_rst_mm 7676056
Error: rst_mm: -1073737728
Error: make_baby: -1073737728
Error: rst_clone_children
Error: make_baby: -1073737728
Error: rst_clone_children
Container start failed
Stopping container ...
Container was stopped
Container is unmounted
-----------------------------------------------
```

I haven't figured out the exact rss size that will cause this but the threshold is somewhere around 2GB.

Note that I can have 8 gig resident and successfully checkpoint/resume as long as no single process has an RSS of more than about 2G. (It does take quite a while to write the checkpoint

file.... Any easy way to gzip on the fly during write and uncompress on the fly on the read for restore?)

The following C program can cause the issue...

```
------------------------------------------------
#include <stdlib.h>
main()
{

   char* m = malloc(8000000000);
   long i = 0;

/* keep entire  block resident if possible*/
while (1){
   for (i=0; i< 8000000000; i++)
     m[i]=1;
  }

}
------------------------------------------------
```

Though I originally tripped over this issue with much more complicated usage.

This is supposed to work right? Is there something obvious I'm missing? Does anyone have any pointers on how to debug this?

As far as I can tell reading vzctl and kernel sources there isn't a good way for EBADFD to be getting returned from the ioctl to /proc/cpt - perhaps it's getting sent back up though an 'effor-fd' ioctl setting but I've not had a chance to dig that far yet.

I also note that malformed programs can indefinitely delay checkpointing from working... - do a vfork and then don't exec (infinite loop...) - but that's a different fish to fry... :)