Subject: Checkpoint/Restart mini-summit
Posted by Daniel Lezcano on Tue, 15 Jul 2008 10:49:45 GMT
View Forum Message <> Reply to Message

Hi all,

Here is a proposition a more detailed agenda for the checkpoint/restart
mini-summit. If everybody is ok with it, I will update the wiki.

Comments are welcome :)

Thanks
 -- Daniel

===

The Checkpoint/restart is a very big topic and the time at the
mini-summit is short, so I propose a list of document pointers to be
read before the mini-summit, so we can address the checkpoint/restart
topic directly and save precious time :)

* Documentation
   * Zap : www.ncl.cs.columbia.edu/publications/usenix2007_fordist.pdf
   * Metacluster : lxc.sourceforge.net/doc/ols2006/lxc-ols2006.pdf
   * OpenVZ : http://wiki.openvz.org/Checkpointing_and_live_migration
   * Checkpoint/Restart technology :
http://en.wikipedia.org/wiki/Application_checkpointing
   * Virtual Servers and Checkpoint/Restart in Mainstream Linux : Sigops
document

This section is about how to prepare the kernel to implement the
checkpoint/restart

* Preparing the kernel internals
   * Identifying the kernel subsystems
   * Identifying the process resources
   * Identifying the frameworks for the CR
   * Identifying the pieces to target first

Actually, one of the big interrogation is how we transmit the internal
state to and from the kernel. There are some little patches doing the
checkpoint/restart, taking into account a small part of the kernel
resources. Some were made through netlink, others via /proc, others
directly with a syscall. There were solutions proposed in the

containers mailing list to use a core dump like file, or a CR
filesystem. This section is to discuss about that.

* Passing the kernel internal state to/from userspace
  * coredump like file
  * swap per container
  * netlinks
  * CR filesystem
  * army of different call for the CR (proc, existing syscalls, ...)

The following sections addresse the checkpoint/restart itself which
can be split into three parts: the quescient point, the checkpoint and
the restart.

* Checkpointing / Restarting

  * Reaching a quescient point
    * for the network
    * for the processes
    * for the asynchronous IO

  * Checkpoint
    * Preinstalled checkpoint signal handler ?
    * syscall ?
    * tar of a CR filesystem ?
    * monolithic ?

    * Dumping processes hierarchy
    * Identifying the kernel resource dependencies
    * Dumping system wide resources (per namespace ?)
    * Dumping process wide resources (from process context ?)
      (Memory is in between system and process resource)

  * Restarting
    * New binary format handler ?
    * Identifying the kernel resource dependencies
    * Restoring the processes hierarchy
    * Restoring system wide resources
    * Restoring process wide resources

There is a posix draft, 1003.1m, which specify a CR semantic. This can
be interesting to take it into account and provide an user API based
on this specification so we can keep in mind this when we implement
the CR in the kernel. I was not able to find the posix draft itself
but the man of the CR IRIX implementation sticks to this
specification.

* Determining the userspace API
    * Posix 1003.1m (implementation in IRIX) ?

http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi/0650/bks/SGI_Admin/CPR_OG/sgi_html/ch03.html