Subject: Re: [RFC][PATCH 0/2] pipe: checkpoint and restart for pipe; brief description and test interface
Posted by Cedric Le Goater on Wed, 31 Jan 2007 15:28:22 GMT
View Forum Message <> Reply to Message

Masahiko Takahashi wrote:
>
> I would like to post a small patch that implements pipe checkpoint
> and restart functionality.

so do you have some test programs that uses it ? It would most
interesting to see how all fits together.

It seems to me that we need to think about the whole framework of
file checkpoint and restart before anything else. sure pipes are
kinda a special case because they are created by pairs (just
like socketpair).

but all files have things in common, they have an fd which requires
to be the same at restart. they can be shared between processes
to start with which needs to be optimized at checkpoint and taken
into account. they have flags, etc.

then, some have pending data, like socket and pipes.

did you start to work on such a framework ?

> The patch supports pipe buffers used with splice/vmsplice systemcall.
> Usually, spliced buffers' pages are in memory cache because their data
> are read from an filesystem. However the patch doesn't treat buffers'
> pages as cached memory when restoring. If trying to treat as cached
> memory, each buffer's page should be tagged with its filename when
> checkpointing. This is not a difficult implementation, but I'm still
> wondering this is worth implementing...
>
> [RFC][PATCH 1/2] pipe: header file and removal of dupfd()'s static
> [RFC][PATCH 2/2] pipe: checkpoint and restart entity
>
> The patch is for kernel 2.6.19.2 and tested with pipe, splice and
> vmsplice systemcalls.
>
> Interfaces:
> int ckpoint_pipe(struct file *filp, struct cr_pipe *img)
> A pipe (specified by "filp") and its information/buffers are
> checkpointed to image buffer "img". (img must has enough memory)
>
> int restore_pipe(int *fd, struct cr_pipe *img)
> The checkpointed pipe information/buffers "img" is restored

> to kernel and installed to the current process's file
> descriptors "fd". fd[0] is pipe for read and fd[1] for write.
>
> TODO:
> - Non-blocking (signal delivery) and waiting process list.
> - Parameter checks (image buffer size, etc).
> - Error checks.
> - Support file descriptor sharing.
> - More tests.
>
>
> The attached source is a test/debug interface for pipe checkpoint and
> restart on Linux 2.6.19.2. I borrowed this from OpenVZ patch with some
> naming and other changes.
>
>
> Masahiko.
>
>
>
> ---
>
> --- /dev/null 2006-05-22 07:25:23.000000000 -0700
> +++ linux-2.6.19.2/fs/pipe-cr-debug.c 2007-01-22 15:39:05.000000000 -0800
> @@ -0,0 +1,115 @@
> +#include <linux/mm.h>
> +#include <linux/file.h>
> +#include <linux/poll.h>
> +#include <linux/slab.h>
> +#include <linux/module.h>
> +#include <linux/init.h>
> +#include <linux/fs.h>
> +#include <linux/mount.h>
> +#include <linux/pipe_fs_i.h>
> +#include <linux/uio.h>
> +
> +#include <linux/device.h>
> +
> +#include <asm/uaccess.h>
> +#include <asm/ioctls.h>
> +
> +#include <linux/cr.h>
> +
> +static struct class *crctl_class;
> +
> +#define CRCTL_MAJOR 200
> +#define CRCTL_NAME "crctl"
> +

```
> +extern int ckpoint_pipe(struct file *filp, struct cr_pipe *img);
> +extern int restore_pipe(int *fd, struct cr_pipe *img);
> +
> +
> +static int cr_open(struct inode *inode, struct file *filp)
> +{
> + return 0;
> +}
> +
> +static int cr_release(struct inode *inode, struct file *filp)
> +{
> + return 0;
> +}
> +
> +static int crctl_ioctl(struct inode *pino, struct file *filp,
> +    unsigned int cmd, unsigned long arg)
> +{
> + int fd[2];
> + struct mig_arg {
> +  void    *arg1;
> +  void    *arg2;
> + } i;
> +
> + switch(cmd) {
> +  case 246:
> +  {
> +   struct file *pipefp;
> +   int r, fput;
> +
> +   if (copy_from_user (&i, (void __user *) arg,
> +    sizeof(i))) return -EFAULT;
> +   pipefp = fget_light((int) i.arg1, &fput);
> +   r = ckpoint_pipe(pipefp, (struct cr_pipe *) i.arg2);
> +   fput_light(pipefp, fput);
> +   return r;
> +  }
> +  case 245:
> +   if (copy_from_user(&i, (void __user *) arg,
> +    sizeof(i))) return -EFAULT;
> +   if (copy_from_user(fd, (void __user *) i.arg1,
> +    sizeof(int) * 2)) return -EFAULT;
> +   return restore_pipe(fd, (struct cr_pipe *) i.arg2);
> + }
> + return 0;
> +}
> +
> +static struct file_operations crctl_fops = {
> + .open  = cr_open,
```

```
> + .release = cr_release,
> + .ioctl  = crctl_ioctl,
> +};
> +
> +static void __exit crctl_exit(void)
> +{
> + class_device_destroy(crctl_class, MKDEV(CRCTL_MAJOR, 0));
> + class_destroy(crctl_class);
> + unregister_chrdev(CRCTL_MAJOR, CRCTL_NAME);
> +}
> +
> +static int __init crctl_init(void)
> +{
> + int ret;
> + struct class_device *class_err;
> +
> + ret = register_chrdev(CRCTL_MAJOR, CRCTL_NAME, &crctl_fops);
> + if (ret < 0)
> +  goto out;
> +
> + crctl_class = class_create(THIS_MODULE, "crctl");
> + if (IS_ERR(crctl_class)) {
> +  ret = PTR_ERR(crctl_class);
> +  goto out_cleandev;
> + }
> +
> + class_err = class_device_create(crctl_class, NULL,
> +  MKDEV(CRCTL_MAJOR, 0), NULL, CRCTL_NAME);
> + if (IS_ERR(class_err)) {
> +  ret = PTR_ERR(class_err);
> +  goto out_rmclass;
> + }
> + goto out;
> +
> +out_rmclass:
> + class_destroy(crctl_class);
> +out_cleandev:
> + unregister_chrdev(CRCTL_MAJOR, CRCTL_NAME);
> +out:
> + return ret;
> +}
> +
> +fs_initcall(crctl_init);
> +module_exit(crctl_exit);
> --- linux-2.6.19.2/fs/Makefile.original 2007-01-10 11:10:37.000000000 -0800
> +++ linux-2.6.19.2/fs/Makefile 2007-01-22 15:13:59.000000000 -0800
> @@ -10,7 +10,7 @@ obj-y := open.o read_write.o file_table.
>    ioctl.o readdir.o select.o fifo.o locks.o dcache.o inode.o \
```

>   attr.o bad_inode.o file.o filesystems.o namespace.o aio.o \
>   seq_file.o xattr.o libfs.o fs-writeback.o \
> - pnode.o drop_caches.o splice.o sync.o utimes.o
> + pnode.o drop_caches.o splice.o sync.o utimes.o pipe-cr-debug.o
>
> ifeq ($(CONFIG_BLOCK),y)
> obj-y += buffer.o bio.o block_dev.o direct-io.o mpage.o ioprio.o
>
>
> ---
>
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers
>
>
> ---
>

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers