
Subject: Re: [PATCH] cryo: Improve socket support: add UDP and IPv6 - V2
Posted by [serue](#) on Tue, 17 Jun 2008 14:00:51 GMT

[View Forum Message](#) <> [Reply to Message](#)

Quoting Benjamin Thery (benjamin.thery@bull.net):

> I played a bit with inet sockets and cryo. With few modifications I
> managed to "checkpoint-restart" programs that use UDP sockets to listen
> and send data.
>
> This patch improves socket checkpointing in cryo:
> * Save/restore socket family: we can checkpoint IPv6 sockets now.
> * Save/restore socket type: we can checkpoint UDP sockets too.
>
> Changelog:
> -----
> * V2:
> - Add helper issockbound() to improve readability of getsockinfo()
> - Cleaned up a bunch of unneeded casts in getsockinfo()
> - Move socket binding code from restore_sock() to bind_sock()
> * V1:
> - Initial version
>
> Simple test programs for UDP sockets available upon request :)
>
> Regards,
> Benjamin
>
> Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>

Applied, thanks.

Please do send the test programs, and I'll add them under cryo/tests.

-serge

> ---
> cr.c | 93 ++-----
> 1 file changed, 73 insertions(+), 20 deletions(-)
>
> Index: cryodev/cr.c
> =====
> --- cryodev.orig/cr.c
> +++ cryodev/cr.c
> @@ -67,11 +67,15 @@ int pagesize = 0;
>
> typedef struct isockinfo_t {
> int fdnum;
> - int type; /* socket type: UDP, TCP */

```

> + int type; /* socket type: SOCK_STREAM, SOCK_DGRAM */
> + int family; /* socket family: AF_INET, AF_INET6 */
> int mode; /* bind, listen, connect */
> int opt; /* various sockopt */
> int backlog; /* for listen(2) */
> - struct sockaddr_in locaddr; /* for bind(2) */
> + union {
> + struct sockaddr_in saddr4;
> + struct sockaddr_in6 saddr6;
> + } locaddr; /* for bind(2) */
> char tcpstate[TPI_LEN];
> } isockinfo_t;
>
> @@ -177,9 +181,25 @@ int getmaps(pid_t pid, memseg_t **p)
> return nseg;
> }
>
> +static int issockbound(struct sockaddr *sa)
> +{
> + if (sa->sa_family == AF_INET) {
> + struct sockaddr_in *sin = (struct sockaddr_in *) sa;
> + if (sin->sin_addr.s_addr != INADDR_ANY ||
> + sin->sin_port != 0)
> + return 1;
> + } else if (sa->sa_family == AF_INET6) {
> + struct sockaddr_in6 *sin6 = (struct sockaddr_in6 *) sa;
> + if (memcmp(&sin6->sin6_addr, &in6addr_any, 16) != 0 ||
> + sin6->sin6_port != 0)
> + return 1;
> + }
> + return 0;
> +}
> +
> int getsockinfo(pid_t pid, pinfo_t *pi, int num)
> {
> - struct sockaddr name;
> + struct sockaddr_storage name;
> int namelen = (int)sizeof(name), ret = 0;
> int tpi_len = TPI_LEN;
> int flag, flaglen;
> @@ -188,35 +208,46 @@ int getsockinfo(pid_t pid, pinfo_t *pi,
> /* get socketname and see if it is an unix or an inet socket */
> memset(&name, 0, sizeof(name));
> PT_GETSOCKNAME(pid, num, &name, &namelen);
> - t_d(name.sa_family);
> + t_d(name.ss_family);
>
> - if (name.sa_family != (sa_family_t)AF_INET) return ret;

```

```

> + if (name.ss_family != (sa_family_t)AF_INET &&
> +     name.ss_family != (sa_family_t)AF_INET6)
> +     return ret;
>
> if (! (pi->si = (isockinfo_t *)realloc(pi->si, sizeof(isockinfo_t) * (pi->ns+1)))) return -1;
> psi = &pi->si[pi->ns];
> - memset((void *)psi, 0, sizeof(*psi));
> + memset(psi, 0, sizeof(*psi));
>
> //pi->si[pi->ns].fdnum = num;
> psi->fdnum = num;
> + psi->family = name.ss_family;
>
> psi->mode = ISOCK_UNKNOWN;
> - memcpy((void *)&psi->locaddr, (void *)&name, sizeof(name));
> - if (psi->locaddr.sin_addr.s_addr != 0 || psi->locaddr.sin_port != 0) psi->mode |= ISOCK_BIND;
> -
> + memcpy(&psi->locaddr, &name, namelen);
> +
> + if (issockbound((struct sockaddr *)&psi->locaddr))
> +     psi->mode |= ISOCK_BIND;
> +
> + flag = 0;
> + flaglen = sizeof(flag);
> + PT_GETSOCKOPT(pid, num, SOL_SOCKET, SO_TYPE, &flag, &flaglen);
> + if (flag)
> +     psi->type = flag;
> +
> flag = 0;
> - flaglen = (int)sizeof(flag);
> + flaglen = sizeof(flag);
> PT_GETSOCKOPT(pid, num, SOL_SOCKET, SO_REUSEADDR, &flag, &flaglen);
> if (flag) psi->opt |= ISOCK_REUSEADDR;
>
> flag = 0;
> - flaglen = (int)sizeof(flag);
> + flaglen = sizeof(flag);
> PT_GETSOCKOPT(pid, num, SOL_SOCKET, SO_ACCEPTCONN, &flag, &flaglen);
> if (flag) {
>     psi->mode |= ISOCK_LISTEN;
>     //t_d(PT_GETSOCKOPT(pid, num, SOL_TCP, TPI_INFO, &tpi_info, &tpi_info_len));
> }
>
> - namelen = (int)sizeof(name);
> + namelen = sizeof(name);
> if (PT_GETPEERNAME(pid, num, &name, &namelen) == 0) {
>     t_d(psi->mode |= ISOCK_CONNECT);
>     t_d(PT_GETSOCKOPT(pid, num, SOL_TCP, TPI_TCP_STATE, pi->si[pi->ns].tcpstate,

```

```

&tpi_len));
> @@ -859,6 +890,31 @@ error:
>   return -1;
> }
>
> +static void bind_sock(pid_t pid, isockinfo_t *isockinfo)
> +{
> + int len;
> + struct sockaddr *sa;
> + struct sockaddr_in sin;
> + struct sockaddr_in6 sin6;
> +
> + if (isockinfo->family == AF_INET) {
> + len = sizeof(sin);
> + memcpy(&sin, &isockinfo->locaddr.saddr4, len);
> + if (isockinfo->mode & ISOCK_CONNECT)
> + sin.sin6_port = htons(0);
> + sa = (struct sockaddr *) &sin;
> + } else if (isockinfo->family == AF_INET6) {
> + len = sizeof(sin6);
> + memcpy(&sin6, &isockinfo->locaddr.saddr6, len);
> + if (isockinfo->mode & ISOCK_CONNECT)
> + sin.sin_port = htons(0);
> + sa = (struct sockaddr *) &sin6;
> + } else
> + ERROR("Unsupported socket family: %d\n",
> + isockinfo->family);
> + t_d(PT_BIND(pid, isockinfo->fdnum, sa, len));
> +}
> +
> int restore_sock(int fd, pid_t pid)
> {
> char item[64];
> @@ -873,21 +929,18 @@ int restore_sock(int fd, pid_t pid)
> else ITEM_SET(isockinfo, isockinfo_t);
> else break; // unknown item
>
> - t_d(sock = PT_SOCKET(pid, AF_INET, SOCK_STREAM, 0));
> + t_d(sock = PT_SOCKET(pid, isockinfo->family, isockinfo->type, 0));
> if (sock < 0) {
> ERROR("PT_SOCKET(%d, %d, %d, %d) errno=%d: %s",
> - pid, AF_INET, SOCK_STREAM, 0, sock, strerror(-sock));
> + pid, isockinfo->family, isockinfo->type, 0,
> + sock, strerror(-sock));
> }
> t_d(PT_DUP2(pid, sock, isockinfo->fdnum));
> if (sock != isockinfo->fdnum) t_d(PT_CLOSE(pid, sock));
>

```

```
> - if (isockinfo->mode & ISOCK_BIND) {  
> -   struct sockaddr_in sin;  
> +   if (isockinfo->mode & ISOCK_BIND)  
> +     bind_sock(pid, isockinfo);  
>  
> -   memcpy((void *)&sin, (void *)&isockinfo->locaddr, sizeof(sin));  
> -   if (isockinfo->mode & ISOCK_CONNECT) sin.sin_port = htons(0);  
> -   t_d(PT_BIND(pid, isockinfo->fdnum, &sin, sizeof(sin)));  
> - }  
>   if (isockinfo->mode & ISOCK_LISTEN) {  
>     t_d(PT_LISTEN(pid, isockinfo->fdnum, 64)); //FIXME: get backlog from TPI_INFO  
>   }  
>  
> --
```

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