Subject: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by Stanislav Kinsbursky on Thu, 18 Oct 2012 10:23:22 GMT
View Forum Message <> Reply to Message

This patch is required for checkpoint/restore in userspace.
IOW, c/r requires some way to get all pending IPC messages without deleting
them from the queue (checkpoint can fail and in this case tasks will be resumed,
so queue have to be valid).
To achive this, new operation flag MSG_COPY for sys_msgrcv() system call was
introduced. If this flag was specified, then mtype is interpreted as number of
the message to copy.
If MSG_COPY is set, then kernel will allocate dummy message with passed size,
and then use new copy_msg() helper function to copy desired message (instead of
unlinking it from the queue).

Notes:
1) Return -ENOSYS if MSG_COPY is specified, but CONFIG_CHECKPOINT_RESTORE is
not set.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 include/uapi/linux/msg.h | 1 +
 ipc/msg.c | 50 ++--
 ipc/msgutil.c | 38 ++++++++++++++++++++++++++++++++++++++
 ipc/util.h | 1 +
 4 files changed, 88 insertions(+), 2 deletions(-)

```
diff --git a/include/uapi/linux/msg.h b/include/uapi/linux/msg.h
index 76999c9..c1af84a 100644
--- a/include/uapi/linux/msg.h
+++ b/include/uapi/linux/msg.h
@@ -11,6 +11,7 @@
 /* msgrcv options */
 #define MSG_NOERROR     010000  /* no error if message is too big */
 #define MSG_EXCEPT      020000  /* recv any msg except of specified type.*/
+#define MSG_COPY        040000  /* copy (not remove) all queue messages */

 /* Obsolete, used only for backwards compatibility and libc5 compiles */
 struct msqid_ds {
diff --git a/ipc/msg.c b/ipc/msg.c
index 028ab87..a908529 100644
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -789,19 +789,48 @@ long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long msgtyp,
  struct msg_msg *msg;
  int mode;
  struct ipc_namespace *ns;
+#ifdef CONFIG_CHECKPOINT_RESTORE
```

```
+ struct msg_msg *copy = NULL;
+ unsigned long copy_number = 0;
+#endif

  if (msqid < 0 || (long) bufsz < 0)
   return -EINVAL;
+ if (msgflg & MSG_COPY) {
+#ifdef CONFIG_CHECKPOINT_RESTORE
+
+  if (msgflg & MSG_COPY) {
+   copy_number = msgtyp;
+   msgtyp = 0;
+  }
+
+  /*
+   * Create dummy message to copy real message to.
+   */
+  copy = load_msg(buf, bufsz);
+  if (IS_ERR(copy))
+   return PTR_ERR(copy);
+  copy->m_ts = bufsz;
+#else
+  return -ENOSYS;
+#endif
+ }
  mode = convert_mode(&msgtyp, msgflg);
  ns = current->nsproxy->ipc_ns;

  msq = msg_lock_check(ns, msqid);
- if (IS_ERR(msq))
+ if (IS_ERR(msq)) {
+#ifdef CONFIG_CHECKPOINT_RESTORE
+  if (msgflg & MSG_COPY)
+   free_msg(copy);
+#endif
   return PTR_ERR(msq);
+ }

  for (;;) {
   struct msg_receiver msr_d;
   struct list_head *tmp;
+  long msg_counter = 0;

   msg = ERR_PTR(-EACCES);
   if (ipcperms(ns, &msq->q_perm, S_IRUGO))
@@ -821,8 +850,16 @@ long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long msgtyp,
     if (mode == SEARCH_LESSEQUAL &&
       walk_msg->m_type != 1) {
```

```
      msgtyp = walk_msg->m_type - 1;
+#ifdef CONFIG_CHECKPOINT_RESTORE
+    } else if (msgflg & MSG_COPY) {
+     if (copy_number == msg_counter) {
+      msg = copy_msg(walk_msg, copy);
+      break;
+     }
+#endif
     } else
      break;
+    msg_counter++;
    }
    tmp = tmp->next;
   }
@@ -835,6 +872,10 @@ long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long msgtyp,
    msg = ERR_PTR(-E2BIG);
    goto out_unlock;
   }
+#ifdef CONFIG_CHECKPOINT_RESTORE
+   if (msgflg & MSG_COPY)
+    goto out_unlock;
+#endif
   list_del(&msg->m_list);
   msq->q_qnum--;
   msq->q_rtime = get_seconds();
@@ -918,8 +959,13 @@ out_unlock:
    break;
   }
  }
- if (IS_ERR(msg))
+ if (IS_ERR(msg)) {
+#ifdef CONFIG_CHECKPOINT_RESTORE
+  if (msgflg & MSG_COPY)
+   free_msg(copy);
+#endif
   return PTR_ERR(msg);
+ }

  bufsz = msg_handler(buf, msg, bufsz);
  free_msg(msg);
diff --git a/ipc/msgutil.c b/ipc/msgutil.c
index 26143d3..b281f5c 100644
--- a/ipc/msgutil.c
+++ b/ipc/msgutil.c
@@ -100,7 +100,45 @@ out_err:
  free_msg(msg);
  return ERR_PTR(err);
 }
```

```
+#ifdef CONFIG_CHECKPOINT_RESTORE
+struct msg_msg *copy_msg(struct msg_msg *src, struct msg_msg *dst)
+{
+ struct msg_msgseg *dst_pseg, *src_pseg;
+ int len = src->m_ts;
+ int alen;
+
+ BUG_ON(dst == NULL);
+ if (src->m_ts > dst->m_ts)
+  return ERR_PTR(-EINVAL);
+
+ alen = len;
+ if (alen > DATALEN_MSG)
+  alen = DATALEN_MSG;
+
+ dst->next = NULL;
+ dst->security = NULL;

+ memcpy(dst + 1, src + 1, alen);
+
+ len -= alen;
+ dst_pseg = dst->next;
+ src_pseg = src->next;
+ while (len > 0) {
+  alen = len;
+  if (alen > DATALEN_SEG)
+   alen = DATALEN_SEG;
+  memcpy(dst_pseg + 1, src_pseg + 1, alen);
+  dst_pseg = dst_pseg->next;
+  len -= alen;
+  src_pseg = src_pseg->next;
+ }
+
+ dst->m_type = src->m_type;
+ dst->m_ts = src->m_ts;
+
+ return dst;
+}
+#endif
 int store_msg(void __user *dest, struct msg_msg *msg, int len)
 {
  int alen;
diff --git a/ipc/util.h b/ipc/util.h
index 271bded..027f507 100644
--- a/ipc/util.h
+++ b/ipc/util.h
@@ -142,6 +142,7 @@ int ipc_parse_version (int *cmd);
```

```
 extern void free_msg(struct msg_msg *msg);
 extern struct msg_msg *load_msg(const void __user *src, int len);
+extern struct msg_msg *copy_msg(struct msg_msg *src, struct msg_msg *dst);
 extern int store_msg(void __user *dest, struct msg_msg *msg, int len);

 extern void recompute_msgmni(struct ipc_namespace *);
```

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by Michael Kerrisk (man- on Thu, 18 Oct 2012 10:39:09 GMT
View Forum Message <> Reply to Message

On Thu, Oct 18, 2012 at 12:23 PM, Stanislav Kinsbursky
<skinsbursky@parallels.com> wrote:
> This patch is required for checkpoint/restore in userspace.
> IOW, c/r requires some way to get all pending IPC messages without deleting
> them from the queue (checkpoint can fail and in this case tasks will be resumed,
> so queue have to be valid).
> To achive this, new operation flag MSG_COPY for sys_msgrcv() system call was
> introduced. If this flag was specified, then mtype is interpreted as number of
> the message to copy.
> If MSG_COPY is set, then kernel will allocate dummy message with passed size,
> and then use new copy_msg() helper function to copy desired message (instead of
> unlinking it from the queue).
>
> Notes:
> 1) Return -ENOSYS if MSG_COPY is specified, but CONFIG_CHECKPOINT_RESTORE is
> not set.

Stanislav,

A naive question, because I have not followed C/R closely. How do you
deal with the case that other processes may be reading from the queue?
(Or is that disabled during checkpointing?)

Thanks,

Michael

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by Stanislav Kinsbursky on Thu, 18 Oct 2012 11:02:32 GMT
View Forum Message <> Reply to Message

> On Thu, Oct 18, 2012 at 12:23 PM, Stanislav Kinsbursky
> <skinsbursky@parallels.com> wrote:

>> This patch is required for checkpoint/restore in userspace.
>> IOW, c/r requires some way to get all pending IPC messages without deleting
>> them from the queue (checkpoint can fail and in this case tasks will be resumed,
>> so queue have to be valid).
>> To achive this, new operation flag MSG_COPY for sys_msgrcv() system call was
>> introduced. If this flag was specified, then mtype is interpreted as number of
>> the message to copy.
>> If MSG_COPY is set, then kernel will allocate dummy message with passed size,
>> and then use new copy_msg() helper function to copy desired message (instead of
>> unlinking it from the queue).
>>
>> Notes:
>> 1) Return -ENOSYS if MSG_COPY is specified, but CONFIG_CHECKPOINT_RESTORE is
>> not set.
>
> Stanislav,
>
> A naive question, because I have not followed C/R closely. How do you
> deal with the case that other processes may be reading from the queue?
> (Or is that disabled during checkpointing?)
>

To be honest, in this case behaviour in user-space is unpredictable.
I.e. if you have, for example, 5 messages in queue and going to peek them all,
and another process is reading the queue in the same time, then, most probably,
you won't peek all the 5 and receive ENOMSG.
But this case can be easily handled by user-space application (number of
messages in queue can be discovered before peeking).

Note, that in CRIU IPC resources will be collected when all processes to migrate
are frozen.

> Thanks,
>
> Michael
>

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by Michael Kerrisk (man- on Thu, 18 Oct 2012 11:18:17 GMT
View Forum Message <> Reply to Message

On Thu, Oct 18, 2012 at 1:02 PM, Stanislav Kinsbursky

<skinsbursky@parallels.com> wrote:

>
>> On Thu, Oct 18, 2012 at 12:23 PM, Stanislav Kinsbursky
>> <skinsbursky@parallels.com> wrote:
>>>
>>> This patch is required for checkpoint/restore in userspace.
>>> IOW, c/r requires some way to get all pending IPC messages without
>>> deleting
>>> them from the queue (checkpoint can fail and in this case tasks will be
>>> resumed,
>>> so queue have to be valid).
>>> To achive this, new operation flag MSG_COPY for sys_msgrcv() system call
>>> was
>>> introduced. If this flag was specified, then mtype is interpreted as
>>> number of
>>> the message to copy.
>>> If MSG_COPY is set, then kernel will allocate dummy message with passed
>>> size,
>>> and then use new copy_msg() helper function to copy desired message
>>> (instead of
>>> unlinking it from the queue).
>>>
>>> Notes:
>>> 1) Return -ENOSYS if MSG_COPY is specified, but CONFIG_CHECKPOINT_RESTORE
>>> is
>>> not set.
>>
>>
>> Stanislav,
>>
>> A naive question, because I have not followed C/R closely. How do you
>> deal with the case that other processes may be reading from the queue?
>> (Or is that disabled during checkpointing?)
>>
>
> To be honest, in this case behaviour in user-space is unpredictable.
> I.e. if you have, for example, 5 messages in queue and going to peek them
> all, and another process is reading the queue in the same time, then, most
> probably, you won't peek all the 5 and receive ENOMSG.
> But this case can be easily handled by user-space application (number of
> messages in queue can be discovered before peeking).
>
> Note, that in CRIU IPC resources will be collected when all processes to
> migrate are frozen.

Perhaps I am missing something fundamental, but how can C/R sanely do
anything at all here?

For example, suppose a process reads and processes a message after you
read it with MSG_COPY. Then the remaining messages are all shifted by
one position, and you are going to miss reading one of them. IIUC the
idea of MSG_COPY is to allow you to retrieve a copy of all messages in
the list. It sounds like there's no way this can be done reliably. So,
what possible use does the operation have?

Thanks,

Michael

--
Michael Kerrisk
Linux man-pages maintainer; http://www.kernel.org/doc/man-pages/
Author of "The Linux Programming Interface"; http://man7.org/tlpi/

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by Stanislav Kinsbursky on Thu, 18 Oct 2012 11:34:47 GMT
View Forum Message <> Reply to Message

> On Thu, Oct 18, 2012 at 1:02 PM, Stanislav Kinsbursky
> <skinsbursky@parallels.com> wrote:

>>
>>> On Thu, Oct 18, 2012 at 12:23 PM, Stanislav Kinsbursky
>>> <skinsbursky@parallels.com> wrote:
>>>>
>>>> This patch is required for checkpoint/restore in userspace.
>>>> IOW, c/r requires some way to get all pending IPC messages without
>>>> deleting
>>>> them from the queue (checkpoint can fail and in this case tasks will be
>>>> resumed,
>>>> so queue have to be valid).
>>>> To achive this, new operation flag MSG_COPY for sys_msgrcv() system call
>>>> was
>>>> introduced. If this flag was specified, then mtype is interpreted as
>>>> number of
>>>> the message to copy.
>>>> If MSG_COPY is set, then kernel will allocate dummy message with passed
>>>> size,
>>>> and then use new copy_msg() helper function to copy desired message
>>>> (instead of
>>>> unlinking it from the queue).
>>>>

>>>> Notes:
>>>> 1) Return -ENOSYS if MSG_COPY is specified, but CONFIG_CHECKPOINT_RESTORE
>>>> is
>>>> not set.
>>>
>>>
>>> Stanislav,
>>>
>>> A naive question, because I have not followed C/R closely. How do you
>>> deal with the case that other processes may be reading from the queue?
>>> (Or is that disabled during checkpointing?)
>>>
>>
>> To be honest, in this case behaviour in user-space is unpredictable.
>> I.e. if you have, for example, 5 messages in queue and going to peek them
>> all, and another process is reading the queue in the same time, then, most
>> probably, you won't peek all the 5 and receive ENOMSG.
>> But this case can be easily handled by user-space application (number of
>> messages in queue can be discovered before peeking).
>>
>> Note, that in CRIU IPC resources will be collected when all processes to
>> migrate are frozen.
>
> Perhaps I am missing something fundamental, but how can C/R sanely do
> anything at all here?
>
> For example, suppose a process reads and processes a message after you
> read it with MSG_COPY. Then the remaining messages are all shifted by
> one position, and you are going to miss reading one of them. IIUC the
> idea of MSG_COPY is to allow you to retrieve a copy of all messages in
> the list. It sounds like there's no way this can be done reliably. So,
> what possible use does the operation have?
>

First of all, this problem exist as is regardless to C/R feature or this patch
set. If you share some resource (like message queue in this particular case)
system-wide, then any process A can read out a message, which was send by
process B to process C. So, when processes uses IPC message queues, they should
be designed to handle such failures.

Second, it's up to user-space how to handle such things. It's implied, that
user, trying to migrate some process, holding one end of queue, will also
migrate another process, holding second end.

Third, there is IPC namespace, which isolates IPC objects. It can be used for
safe migration of process tree.

> Thanks,

>
> Michael
>
>

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by ebiederm on Thu, 18 Oct 2012 11:52:48 GMT
View Forum Message <> Reply to Message

Stanislav Kinsbursky <skinsbursky@parallels.com> writes:

> First of all, this problem exist as is regardless to C/R feature or this patch
> set. If you share some resource (like message queue in this particular case)
> system-wide, then any process A can read out a message, which was send by
> process B to process C. So, when processes uses IPC message queues, they should
> be designed to handle such failures.
>
> Second, it's up to user-space how to handle such things. It's implied, that
> user, trying to migrate some process, holding one end of queue, will also
> migrate another process, holding second end.
>
> Third, there is IPC namespace, which isolates IPC objects. It can be used for
> safe migration of process tree.

This does raise an interesting question.

What is the point of the message copy feature? It appears to be simply
an optimization and not needed to actually perform the
checkpoint/restart. If you are going to restart the processes you can
read all of the messages and then write all of the messages back before
you restart the processes.

Eric

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by Michael Kerrisk (man- on Thu, 18 Oct 2012 11:55:07 GMT
View Forum Message <> Reply to Message

>>>> A naive question, because I have not followed C/R closely. How do you
>>>> deal with the case that other processes may be reading from the queue?

>>>> (Or is that disabled during checkpointing?)
>>>>
>>>
>>> To be honest, in this case behaviour in user-space is unpredictable.
>>> I.e. if you have, for example, 5 messages in queue and going to peek them
>>> all, and another process is reading the queue in the same time, then,
>>> most
>>> probably, you won't peek all the 5 and receive ENOMSG.
>>> But this case can be easily handled by user-space application (number of
>>> messages in queue can be discovered before peeking).
>>>
>>> Note, that in CRIU IPC resources will be collected when all processes to
>>> migrate are frozen.
>>
>>
>> Perhaps I am missing something fundamental, but how can C/R sanely do
>> anything at all here?
>>
>> For example, suppose a process reads and processes a message after you
>> read it with MSG_COPY. Then the remaining messages are all shifted by
>> one position, and you are going to miss reading one of them. IIUC the
>> idea of MSG_COPY is to allow you to retrieve a copy of all messages in
>> the list. It sounds like there's no way this can be done reliably. So,
>> what possible use does the operation have?
>>
>
> First of all, this problem exist as is regardless to C/R feature or this
> patch set. If you share some resource (like message queue in this particular
> case) system-wide, then any process A can read out a message, which was send
> by process B to process C. So, when processes uses IPC message queues, they
> should be designed to handle such failures.
>
> Second, it's up to user-space how to handle such things. It's implied, that
> user, trying to migrate some process, holding one end of queue, will also
> migrate another process, holding second end.
>
> Third, there is IPC namespace, which isolates IPC objects. It can be used
> for safe migration of process tree.

Is there somewhere a *detailed* description of how this feature would
be used? Lacking that, it's really hard to see how anything sane and
reliable can be done with MSG_COPY.

--
Michael Kerrisk
Linux man-pages maintainer; http://www.kernel.org/doc/man-pages/
Author of "The Linux Programming Interface"; http://man7.org/tlpi/

>>>>> A naive question, because I have not followed C/R closely. How do you
>>>>> deal with the case that other processes may be reading from the queue?
>>>>> (Or is that disabled during checkpointing?)
>>>>>
>>>>
>>>> To be honest, in this case behaviour in user-space is unpredictable.
>>>> I.e. if you have, for example, 5 messages in queue and going to peek them
>>>> all, and another process is reading the queue in the same time, then,
>>>> most
>>>> probably, you won't peek all the 5 and receive ENOMSG.
>>>> But this case can be easily handled by user-space application (number of
>>>> messages in queue can be discovered before peeking).
>>>>
>>>> Note, that in CRIU IPC resources will be collected when all processes to
>>>> migrate are frozen.
>>>
>>>
>>> Perhaps I am missing something fundamental, but how can C/R sanely do
>>> anything at all here?
>>>
>>> For example, suppose a process reads and processes a message after you
>>> read it with MSG_COPY. Then the remaining messages are all shifted by
>>> one position, and you are going to miss reading one of them. IIUC the
>>> idea of MSG_COPY is to allow you to retrieve a copy of all messages in
>>> the list. It sounds like there's no way this can be done reliably. So,
>>> what possible use does the operation have?
>>>
>>
>> First of all, this problem exist as is regardless to C/R feature or this
>> patch set. If you share some resource (like message queue in this particular
>> case) system-wide, then any process A can read out a message, which was send
>> by process B to process C. So, when processes uses IPC message queues, they
>> should be designed to handle such failures.
>>
>> Second, it's up to user-space how to handle such things. It's implied, that
>> user, trying to migrate some process, holding one end of queue, will also
>> migrate another process, holding second end.
>>
>> Third, there is IPC namespace, which isolates IPC objects. It can be used
>> for safe migration of process tree.
>
> Is there somewhere a *detailed* description of how this feature would
> be used? Lacking that, it's really hard to see how anything sane and
> reliable can be done with MSG_COPY.

>

These patches are used by CRIU already.
So, you can have a look at the CRIU source code:

http://git.criu.org/?p=crtools.git
;a=blob;f=ipc_ns.c;h=9e259fefcfc04ec0556bb722921545552e1c69f 3;hb=HEAD

Sanity and reliability on the level you are talking about can be achieved, only
if you'll freeze all message users before peeking.

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by Michael Kerrisk (man- on Thu, 18 Oct 2012 12:09:09 GMT
View Forum Message <> Reply to Message

>> Is there somewhere a *detailed* description of how this feature would
>> be used? Lacking that, it's really hard to see how anything sane and
>> reliable can be done with MSG_COPY.
>>
>
> These patches are used by CRIU already.
> So, you can have a look at the CRIU source code:
>
> http://git.criu.org/?p=crtools.git
;a=blob;f=ipc_ns.c;h=9e259fefcfc04ec0556bb722921545552e1c69f 3;hb=HEAD
>
> Sanity and reliability on the level you are talking about can be achieved,
> only if you'll freeze all message users before peeking.

Okay -- that's the piece I was looking for. Thanks.

In this scenario, how do you find all of the message users? Or do you
simply ensure that everything is frozen beforehand?

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by ebiederm on Thu, 18 Oct 2012 12:20:54 GMT
View Forum Message <> Reply to Message

"Michael Kerrisk (man-pages)" <mtk.manpages@gmail.com> writes:

>>> Is there somewhere a *detailed* description of how this feature would

>>> be used? Lacking that, it's really hard to see how anything sane and
>>> reliable can be done with MSG_COPY.
>>>
>>
>> These patches are used by CRIU already.
>> So, you can have a look at the CRIU source code:
>>
>> http://git.criu.org/?p=crtools.git
;a=blob;f=ipc_ns.c;h=9e259fefcfc04ec0556bb722921545552e1c69f 3;hb=HEAD
>>
>> Sanity and reliability on the level you are talking about can be achieved,
>> only if you'll freeze all message users before peeking.
>
> Okay -- that's the piece I was looking for. Thanks.
>
> In this scenario, how do you find all of the message users? Or do you
> simply ensure that everything is frozen beforehand?

The general design is a container is started with a fresh set of
namespaces and then the entire is frozen using the process freezer
control group. With all of the userspace process frozen checkpoint then
happens.

Eric

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by Stanislav Kinsbursky on Thu, 18 Oct 2012 12:41:34 GMT
View Forum Message <> Reply to Message

>>> Is there somewhere a *detailed* description of how this feature would
>>> be used? Lacking that, it's really hard to see how anything sane and
>>> reliable can be done with MSG_COPY.
>>>
>>
>> These patches are used by CRIU already.
>> So, you can have a look at the CRIU source code:
>>
>> http://git.criu.org/?p=crtools.git
;a=blob;f=ipc_ns.c;h=9e259fefcfc04ec0556bb722921545552e1c69f 3;hb=HEAD
>>
>> Sanity and reliability on the level you are talking about can be achieved,
>> only if you'll freeze all message users before peeking.
>
> Okay -- that's the piece I was looking for. Thanks.
>
> In this scenario, how do you find all of the message users?

It looks like there is no way to how find these users.
But I don't really think, that this is necessary.
I.e. nothing protects the queue from reading by some alien (not expected)
process in real life.

> Or do you simply ensure that everything is frozen beforehand?
>

In this particular case - yes.

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by Stanislav Kinsbursky on Thu, 18 Oct 2012 14:40:01 GMT
View Forum Message <> Reply to Message

> Stanislav Kinsbursky <skinsbursky@parallels.com> writes:
>
>> First of all, this problem exist as is regardless to C/R feature or this patch
>> set. If you share some resource (like message queue in this particular case)
>> system-wide, then any process A can read out a message, which was send by
>> process B to process C. So, when processes uses IPC message queues, they should
>> be designed to handle such failures.
>>
>> Second, it's up to user-space how to handle such things. It's implied, that
>> user, trying to migrate some process, holding one end of queue, will also
>> migrate another process, holding second end.
>>
>> Third, there is IPC namespace, which isolates IPC objects. It can be used for
>> safe migration of process tree.
>
> This does raise an interesting question.
>
> What is the point of the message copy feature? It appears to be simply
> an optimization and not needed to actually perform the
> checkpoint/restart. If you are going to restart the processes you can
> read all of the messages and then write all of the messages back before
> you restart the processes.
>

It's not just an optimisation.
If crtools will fail (with SIGSEGV, for instance), then queue will be empty.

> Eric
>

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by ebiederm on Fri, 19 Oct 2012 00:52:51 GMT
View Forum Message <> Reply to Message

Stanislav Kinsbursky <skinsbursky@parallels.com> writes:

> It's not just an optimisation.
> If crtools will fail (with SIGSEGV, for instance), then queue will be empty.

Regardless of what you call the benefit of this enhancement, this
enhancement is not required to implement checkpoint/restart.

For reliability/restartability I suspect a simple enqueue/dequeue loop
over each message in the queue would be nearly as proof against SIGSEGV
and other failures.

So since all of these changes are enhancements we need to know what we
are getting, over just sticking with the existing interfaces.

Unless there is a real bottleneck for something to work, I suspect the
direction forward is to make checkpoint and restart work with the
existing kernel interfaces and then revisit that decision when you
actually have a real problem.

Eric

Subject: Re: [PATCH v7 09/10] IPC: message queue copy feature introduced
Posted by Stanislav Kinsbursky on Fri, 19 Oct 2012 07:44:33 GMT
View Forum Message <> Reply to Message

> Stanislav Kinsbursky <skinsbursky@parallels.com> writes:
>
>> It's not just an optimisation.
>> If crtools will fail (with SIGSEGV, for instance), then queue will be empty.
>
> Regardless of what you call the benefit of this enhancement, this

> enhancement is not required to implement checkpoint/restart.
>
> For reliability/restartability I suspect a simple enqueue/dequeue loop
> over each message in the queue would be nearly as proof against SIGSEGV
> and other failures.
>

"Nearly as proof" is not good enough for CRIU quality of service.
Moreover, if crtools will fail in this loop, then not only one message will be
lost, but also the queue messages order will be invalid.

> So since all of these changes are enhancements we need to know what we
> are getting, over just sticking with the existing interfaces.
>
> Unless there is a real bottleneck for something to work, I suspect the
> direction forward is to make checkpoint and restart work with the
> existing kernel interfaces and then revisit that decision when you
> actually have a real problem.
>
> Eric
>

--
Best regards,
Stanislav Kinsbursky