Subject: Re: [PATCH 16/28] [FLAT 1/6] Changes in data structures for flat model
Posted by Pavel Emelianov on Tue, 19 Jun 2007 07:18:17 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> Pavel Emelianov [xemul@openvz.org] wrote:
> | This patch opens the flat model patches.
> |
> | The flat model idea is that struct pid has two numbers. The first one
> | (pid->nr) is a global one and is unique in the system. The second one
> | (pid->vnr) is a virtual pid. It is used on the kernel user boundary only.
>
> This approach duplicates 5 integers and 2 pointers per process for every
> process in the system. While this may not be expensive for processes that
> actually use multiple namespaces, doesn't it waste memory if majority of
> processes exist only in one namespace ?

task_struct alignment allows for it. so does the alignment of signal structure.
and please note that this comes with appropriate ifdefs around. the only problem
is with struct pid, but we're virtualizing it after all!

moreover - two integers and a pointer to the namespace is the minimal set of
fields for pid that is visible from two namespaces...

> |
> | I.e. it is shown to user via getpid(), proc, etc, and when the application
> | passes pid to the kernel to make something with proc, this number is treated
> | as the virtual one and the pid/task is searched in the namespace caller task
> | belongs to.
> |
> |
> |
> | The struct pid must have two numerical values, the pointer to the namespace and
> | additional hlist_node to hash the pid in two hash-tables.
> |
> | The virtual ids are stored on struct task_struct and struct signal together with
> | their global pairs for optimisation.
> |
> | Signed-off-by: Pavel Emelianov <xemul@openvz.org>
> |
> | ---
> |
> | pid.h | 19 ++++++++++++++++++
> | sched.h | 10 +++++++++-
> | 2 files changed, 28 insertions(+), 1 deletion(-)
> |
> | --- ./include/linux/pid.h.flatdatast 2007-06-15 15:08:39.000000000 +0400
> | +++ ./include/linux/pid.h 2007-06-15 15:22:18.000000000 +0400

```
> | @@ -40,12 +40,31 @@ enum pid_type
> |   * processes.
> |   */
> |
> | +/*
> | + * flat pid namespaces.
> | + * each task hash two ids of each type - the global id and the virtual one
> | + * the latter one is used on kernel-user boundary only. i.e. if the kernel
> | + * wants to save the task's pid for some time it may store the global one
> | + * and the use find_pid()/find_task_by_pid() routines as it was used before.
> | + * when an id comes from the usersoace or it must go to user the virtual
> | + * id must be used.
> | + */
> | +
> |  struct pid
> |  {
> |   atomic_t count;
> |   /* Try to keep pid_chain in the same cacheline as nr for find_pid */
> |   int nr;
> |   struct hlist_node pid_chain;
> | +#ifdef CONFIG_PID_NS_FLAT
> | + /*
> | +  * virtual number, the namespace this pid belongs to and the hlist_node
> | +  * to find this pid by vnr in hash
> | +  */
> | + int vnr;
> | + struct pid_namespace *ns;
> | + struct hlist_node vpid_chain;
> | +#endif
> |   /* lists of tasks that use this pid */
> |   struct hlist_head tasks[PIDTYPE_MAX];
> |   struct rcu_head rcu;
> | --- ./include/linux/sched.h.flatdatast 2007-06-15 15:14:33.000000000 +0400
> | +++ ./include/linux/sched.h 2007-06-15 15:19:14.000000000 +0400
> | @@ -482,7 +482,10 @@ struct signal_struct {
> |    pid_t session __deprecated;
> |    pid_t __session;
> |   };
> | -
> | +#ifdef CONFIG_PID_NS_FLAT
> | + pid_t vpgrp;
> | + pid_t vsession;
> | +#endif
> |   /* boolean value for session group leader */
> |   int leader;
> |
> | @@ -944,6 +947,11 @@ struct task_struct {
> |   unsigned did_exec:1;
```

> | pid_t pid;
> | pid_t tgid;
> | +#ifdef CONFIG_PID_NS_FLAT
> | + /* hash the virtual ids as well */
> | + pid_t vpid;
> | + pid_t vtgid;
> | +#endif
> |
> | #ifdef CONFIG_CC_STACKPROTECTOR
> | /* Canary value for the -fstack-protector gcc feature */
>

_____

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