Subject: Re: [RFC][PATCH] introduce task cgroup (#task restrictioon for prevent fork bomb by cgroup)
Posted by Li Zefan on Thu, 05 Jun 2008 05:47:39 GMT
View Forum Message <> Reply to Message

KOSAKI Motohiro wrote:
> Hi
>
> I create new cgroup of number task restriction.
> Please any comments!
>
>
>
> benefit
> ==
> 1. prevent fork bomb.
>
> We already have "/prox/sys/kernel/threads-max".
> but it isn't perfect solution.
> because threads-max prevent any process creation.
> then, System-administrator can't login and restore trouble.
>
> restrict of cgroup is better solution.
> it can prevent fork by network service daemon, but allow fork interactive operation.
>
>
> 2. help implement batch processing
>
> in general, batch environment need support #task restriction.
> my patch help implement it.
>
>
> usage
> ==
>
> # mount -t cgroup -o task none /dev/cgroup
> # mkdir /dev/cgroup/foo
> # cd /dev/cgroup/foo
> # ls
> notify_on_release task.max_tasks task.nr_tasks tasks
> # echo 100 > task.max_tasks
> # fork_bomb 1000 & <- try create 1000 process
> # pgrep fork_bomb|wc -l
> 98
>

You are not handling task migration between groups, i.e.:
 # echo $$ > /cgroup/1/tasks

```
 # echo $$ > /cgroup/2/tasks
```

```
>
> future work
> ==================================
> discussion cgroup guys more.
>
>
> Signed-off-by: KOSAKI Motohiro <kosaki.motohiro@jp.fujitsu.com>
> CC: Li Zefan <lizf@cn.fujitsu.com>
> CC: Paul Menage <menage@google.com>
>
> ---
>  include/linux/cgroup.h        |    5 -
>  include/linux/cgroup_subsys.h |    4
>  init/Kconfig                  |   10 ++
>  kernel/Makefile               |    1
>  kernel/cgroup.c               |   16 +++
>  kernel/cgroup_task.c          |  185 ++++++++++++++++++++++++++++++++++++++++++++
>  kernel/fork.c                 |    5 -
>  7 files changed, 222 insertions(+), 4 deletions(-)
>
> Index: b/include/linux/cgroup.h
> ===================================================================
> --- a/include/linux/cgroup.h
> +++ b/include/linux/cgroup.h
> @@ -27,7 +27,7 @@ extern int cgroup_init(void);
>  extern void cgroup_init_smp(void);
>  extern void cgroup_lock(void);
>  extern void cgroup_unlock(void);
> -extern void cgroup_fork(struct task_struct *p);
> +extern int  cgroup_fork(struct task_struct *p);
>  extern void cgroup_fork_callbacks(struct task_struct *p);
>  extern void cgroup_post_fork(struct task_struct *p);
>  extern void cgroup_exit(struct task_struct *p, int run_callbacks);
> @@ -299,6 +299,7 @@ struct cgroup_subsys {
>       struct cgroup *cgrp, struct task_struct *tsk);
>   void (*attach)(struct cgroup_subsys *ss, struct cgroup *cgrp,
>     struct cgroup *old_cgrp, struct task_struct *tsk);
> + int  (*can_fork)(struct cgroup_subsys *ss, struct task_struct *task);
>   void (*fork)(struct cgroup_subsys *ss, struct task_struct *task);
>   void (*exit)(struct cgroup_subsys *ss, struct task_struct *task);
>   int (*populate)(struct cgroup_subsys *ss,
> @@ -381,7 +382,7 @@ int cgroup_attach_task(struct cgroup *,
>  static inline int cgroup_init_early(void) { return 0; }
>  static inline int cgroup_init(void) { return 0; }
>  static inline void cgroup_init_smp(void) {}
> -static inline void cgroup_fork(struct task_struct *p) {}
```

```
> +static inline int  cgroup_fork(struct task_struct *p) { return 0; }
>  static inline void cgroup_fork_callbacks(struct task_struct *p) {}
>  static inline void cgroup_post_fork(struct task_struct *p) {}
>  static inline void cgroup_exit(struct task_struct *p, int callbacks) {}
> Index: b/init/Kconfig
> ===================================================================
> --- a/init/Kconfig
> +++ b/init/Kconfig
> @@ -289,6 +289,16 @@ config CGROUP_DEBUG
>
>     Say N if unsure
>
> +config CGROUP_TASK
> + bool "Simple number of task accounting cgroup subsystem"
> + depends on CGROUPS && EXPERIMENTAL
> + default n
> + help
> +          Provides a simple number of task accounting cgroup subsystem for
> +   prevent fork bomb.
> +
> +   Say N if unsure
> +
>  config CGROUP_NS
>          bool "Namespace cgroup subsystem"
>          depends on CGROUPS
> Index: b/kernel/Makefile
> ===================================================================
> --- a/kernel/Makefile
> +++ b/kernel/Makefile
> @@ -46,6 +46,7 @@ obj-$(CONFIG_KEXEC) += kexec.o
>  obj-$(CONFIG_COMPAT) += compat.o
>  obj-$(CONFIG_CGROUPS) += cgroup.o
>  obj-$(CONFIG_CGROUP_DEBUG) += cgroup_debug.o
> +obj-$(CONFIG_CGROUP_TASK) += cgroup_task.o
>  obj-$(CONFIG_CPUSETS) += cpuset.o
>  obj-$(CONFIG_CGROUP_NS) += ns_cgroup.o
>  obj-$(CONFIG_UTS_NS) += utsname.o
> Index: b/kernel/cgroup.c
> ===================================================================
> --- a/kernel/cgroup.c
> +++ b/kernel/cgroup.c
> @@ -2719,13 +2719,27 @@ static struct file_operations proc_cgrou
>   * At the point that cgroup_fork() is called, 'current' is the parent
>   * task, and the passed argument 'child' points to the child task.
>   */
> -void cgroup_fork(struct task_struct *child)
> +int cgroup_fork(struct task_struct *child)
>  {
```

> + int i;
> + int ret;
> +
> + for (i = 0; i < CGROUP_SUBSYS_COUNT; i++) {

Maybe you need to add this in cgroup_init_subsys():

-       need_forkexit_callback |= ss->fork || ss->exit;
+   need_forkexit_callback |= ss->fork || ss->exit || ss->can_fork;

and then we can do:

if (need_forkexit_callback) {
 for (i = 0; i < CGROUP_SUBSYS_COUNT; i++) {
  ...
 }
}

> + struct cgroup_subsys *ss = subsys[i];
> + if (ss->can_fork) {
> +   ret = ss->can_fork(ss, child);
> +   if (ret)
> +   return ret;
> + }
> + }
> +
>   task_lock(current);
>   child->cgroups = current->cgroups;
>   get_css_set(child->cgroups);
>   task_unlock(current);
>   INIT_LIST_HEAD(&child->cg_list);
> +
> + return 0;
> }

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