Subject: containers (was Re: -mm merge plans for 2.6.23)
Posted by Srivatsa Vaddagiri on Tue, 10 Jul 2007 10:52:40 GMT
View Forum Message <> Reply to Message

On Tue, Jul 10, 2007 at 01:31:52AM -0700, Andrew Morton wrote:
> cpuset-zero-malloc-revert-the-old-cpuset-fix.patch
> containersv10-basic-container-framework.patch
> containersv10-basic-container-framework-fix.patch
> containersv10-basic-container-framework-fix-2.patch
> containersv10-basic-container-framework-fix-3.patch
> containersv10-basic-container-framework-fix-for-bad-lock-balance-in-containers.patch
> containersv10-example-cpu-accounting-subsystem.patch
> containersv10-example-cpu-accounting-subsystem-fix.patch
> containersv10-add-tasks-file-interface.patch
> containersv10-add-tasks-file-interface-fix.patch
> containersv10-add-tasks-file-interface-fix-2.patch
> containersv10-add-fork-exit-hooks.patch
> containersv10-add-fork-exit-hooks-fix.patch
> containersv10-add-container_clone-interface.patch
> containersv10-add-container_clone-interface-fix.patch
> containersv10-add-procfs-interface.patch
> containersv10-add-procfs-interface-fix.patch
> containersv10-make-cpusets-a-client-of-containers.patch
> containersv10-make-cpusets-a-client-of-containers-whitespace.patch
> containersv10-share-css_group-arrays-between-tasks-with-same-container-memberships.patch
>
containersv10-share-css_group-arrays-between-tasks-with-same-container-memberships-fix.patch
>
containersv10-share-css_group-arrays-between-tasks-with-same-container-memberships-cpuset-zero-malloc-fix-for-new-containers.patch
> containersv10-simple-debug-info-subsystem.patch
> containersv10-simple-debug-info-subsystem-fix.patch
> containersv10-simple-debug-info-subsystem-fix-2.patch
> containersv10-support-for-automatic-userspace-release-agents.patch
> containersv10-support-for-automatic-userspace-release-agents-whitespace.patch
> add-containerstats-v3.patch
> add-containerstats-v3-fix.patch
> update-getdelays-to-become-containerstats-aware.patch
> containers-implement-subsys-post_clone.patch
> containers-implement-namespace-tracking-subsystem-v3.patch
>
> Container stuff. Hold, I guess. I was expecting updates from Paul.

Paul,
Are you working on a new version? I thought it was mostly ready
for mainline.

--
Regards,
vatsa

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

---

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by akpm on Wed, 11 Jul 2007 05:29:42 GMT
View Forum Message <> Reply to Message

On Wed, 11 Jul 2007 10:25:16 +0530 Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:

> On Tue, Jul 10, 2007 at 11:53:19AM -0700, Andrew Morton wrote:
> > On Tue, 10 Jul 2007 11:34:38 -0700
> > "Paul Menage" <menage@google.com> wrote:
> >
> > > Andrew, how about we merge enough of the container framework to
> > > support CFS? Bits we could leave out for now include container_clone()
> > > support and the nsproxy subsystem, fork/exit callback hooks, and
> > > possibly leave cpusets alone for now (which would also mean we could
> > > skip the automatic release-agent stuff). I'm in Tokyo for the Linux
> > > Foundation Japan symposium right now, but I should be able to get the
> > > new patchset to you for Friday afternoon.
> >
> > mm.. Given that you propose leaving bits out for the 2.6.23 merge, and
> > that changes are still pending and that nothing will _use_ the framework in
> > 2.6.23 [...]
>
> Andrew,
> The cpu group scheduler is ready and waiting for the container patches
> in 2.6.23 :)
>
> Here are some options with us:
>
> a. (As Paul says) merge enough of container patches to enable
>   its use with cfs group scheduler (and possibly cpusets?)
>
> b. Enable group scheduling bits in 2.6.23 using the user-id grouping
>   mechanism (aka fair user scheduler). For 2.6.24, we could remove
>   this interface and use Paul's container patches instead. Since this
>   means change of API interface between 2.6.23 and 2.6.24, I don't
>   prefer this option.
>
> c. Enable group scheduling bits only in -mm for now (2.6.23-mmX), using
>   Paul's container patches. I can send you a short patch that hooks up

> cfs group scheduler with Paul's container infrastructure.
>
> If a. is not possible, I would prefer c.
>
> Let me know your thoughts ..

I'm inclined to take the cautious route here - I don't think people will be
dying for the CFS thingy (which I didn't even know about?) in .23, and it's
rather a lot of infrastructure to add for a CPU scheduler configurator
gadget (what does it do, anyway?)

We have plenty of stuff for 2.6.23 already ;)

Is this liveable with??

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

---

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Srivatsa Vaddagiri on Wed, 11 Jul 2007 06:03:09 GMT
View Forum Message <> Reply to Message

On Tue, Jul 10, 2007 at 10:29:42PM -0700, Andrew Morton wrote:
> I'm inclined to take the cautious route here - I don't think people will be
> dying for the CFS thingy (which I didn't even know about?) in .23, and it's
> rather a lot of infrastructure to add for a CPU scheduler configurator
> gadget (what does it do, anyway?)

Hmm ok, if you think the container patches is too early for 2.6.23, fine.
We should definitely target to have it in 2.6.24, by which time I am
thinking the memory rss controller will also be in a good shape.

> We have plenty of stuff for 2.6.23 already ;)
>
> Is this liveable with??

Fine. I will request you to enable group cpu scheduling in
2.6.23-rcX-mmY atleast, so that it gets some amount of testing. The
essential group scheduling bits is already in Linus' tree now (as part
of cfs merge), so what you need in -mm is a slim patch to hook it with Paul's
container infrastructure (which I trust will continue to be in -mm until it goes
mainline). I will send across that slim patch later (to be included in
2.6.23-rc1-mm1 perhaps).

--
Regards,

vatsa

_____

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

---

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Ingo Molnar on Wed, 11 Jul 2007 09:04:23 GMT
View Forum Message <> Reply to Message

* Andrew Morton <akpm@linux-foundation.org> wrote:

> > c. Enable group scheduling bits only in -mm for now (2.6.23-mmX), using
> > Paul's container patches. I can send you a short patch that hooks up
> > cfs group scheduler with Paul's container infrastructure.
> >
> > If a. is not possible, I would prefer c.
> >
> > Let me know your thoughts ..
>
> I'm inclined to take the cautious route here - I don't think people
> will be dying for the CFS thingy (which I didn't even know about?) in
> .23, and it's rather a lot of infrastructure to add for a CPU
> scheduler configurator gadget (what does it do, anyway?)
>
> We have plenty of stuff for 2.6.23 already ;)
>
> Is this liveable with??

another option would be to trivially hook up CONFIG_FAIR_GROUP_SCHED
with cpusets, and to offer CONFIG_FAIR_GROUP_SCHED in the Kconfig,
dependent on CPUSETS and defaulting to off. That would give it a chance
to be tested, benchmarked, etc.

Ingo

_____

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

---

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Paul Jackson on Wed, 11 Jul 2007 09:23:52 GMT
View Forum Message <> Reply to Message

Ingo wrote:

> another option would be to trivially hook up CONFIG_FAIR_GROUP_SCHED
> with cpusets, ...

ah ... you triggered my procmail filter for 'cpuset' ... ;).

What would it mean to hook up CFS with cpusets? I've a pretty
good idea what a cpuset is, but don't know what kind of purpose
you have in mind for such a hook. Could you say a few words to
that? Thanks.

--

I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

_____

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Srivatsa Vaddagiri on Wed, 11 Jul 2007 10:03:23 GMT
View Forum Message <> Reply to Message

On Wed, Jul 11, 2007 at 02:23:52AM -0700, Paul Jackson wrote:
> Ingo wrote:
> > another option would be to trivially hook up CONFIG_FAIR_GROUP_SCHED
> > with cpusets, ...
>
> ah ... you triggered my procmail filter for 'cpuset' ... ;).

:-)

> What would it mean to hook up CFS with cpusets?

CFS is the new cpu scheduler in Linus's tree (http://lwn.net/Articles/241085/).
It has some group scheduling capabilities added i.e the core scheduler
now recognizes the concept of a task-group and providing fair cpu time
to each task-group (in addition to providing fair time to each task in a
group).

The core scheduler however is not concerned with how task groups are formed
and/or how tasks migrate between groups. Thats where a patch like Paul Menage's
container infrastructure comes in hand - to provide a user-interface for
managing task-groups (create/delete task groups, migrate task from one
group to another etc). Whatever the chosen user-interface is, cpu
scheduler needs to know about such task-group creation/destruction,
migration of tasks across groups etc.

Unfortunately, the group-scheduler bits will be ready in 2.6.23 while
Paul Menage's container patches aren't ready for 2.6.23 yet.

So Ingo was proposing we use cpuset as that user interface to manage
task-groups. This will be only for 2.6.23. In 2.6.24, when hopefully Paul
Menage's container patches will be ready and will be merged, the group
cpu scheduler will stop using cpuset as that interface and use the
container infrastructure instead.

If you recall, I have attempted to use cpuset for such an interface in
the past (metered cpusets - see [1]). It brings in some semantic changes
for cpusets, most notably:

- metered cpusets cannot have grand-children
- all cpusets under a metered cpuset need to share the same set
  of cpus.

Is it fine if I introduce these semantic changes, only for 2.6.23 and
only when CONFIG_FAIR_GROUP_SCHED is enabled? This will let the group
cpu scheduler to receive some amount of testing.

The other alternative is to hook up group scheduler with user-id's
(again only for 2.6.23).

> I've a pretty
> good idea what a cpuset is, but don't know what kind of purpose
> you have in mind for such a hook. Could you say a few words to
> that? Thanks.

Reference:

1. http://marc.info/?l=linux-kernel&m=115946525811848&w=2

--
Regards,
vatsa

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

---

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Ingo Molnar on Wed, 11 Jul 2007 10:19:58 GMT
View Forum Message <> Reply to Message

* Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:

> The other alternative is to hook up group scheduler with user-id's
> (again only for 2.6.23).

could you just try this and send an as simple patch as possible? This is actually something that non-container people would be interested in as well. (btw., if this goes into 2.6.23 then we cannot possibly turn it off in 2.6.24, so it must be sane - but per UID task groups are certainly sane, the only question is how to configure the per-UID weight after bootup. [the default after-bootup should be something along the lines of 'same weight for all users, a bit more for root'.]) This would make it possible for users to test that thing. (it would also help X-heavy workloads.)

Ingo

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Paul Jackson on Wed, 11 Jul 2007 11:10:07 GMT
View Forum Message <> Reply to Message

Srivatsa wrote:
> So Ingo was proposing we use cpuset as that user interface to manage
> task-groups. This will be only for 2.6.23.

Good explanation - thanks.

In short, the proposal was to use the task partition defined by cpusets to define CFS task-groups, until the real process containers are available.

Or, I see in the next message, Ingo responding favorably to your alternative, using task uid's to partition the tasks into CFS task-groups.

Yeah, Ingo's preference for using uid's (or gid's ??) sounds right to me - a sustainable API.

Wouldn't want to be adding a cpuset API for a single 2.6.N release.

... gid's -- why not?

--

I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

_____

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Peter Zijlstra on Wed, 11 Jul 2007 11:24:47 GMT
View Forum Message <> Reply to Message

On Wed, 2007-07-11 at 04:10 -0700, Paul Jackson wrote:
> Srivatsa wrote:
> > So Ingo was proposing we use cpuset as that user interface to manage
> > task-groups. This will be only for 2.6.23.
>
> Good explanation - thanks.
>
> In short, the proposal was to use the task partition defined by cpusets
> to define CFS task-groups, until the real process containers are
> available.
>
> Or, I see in the next message, Ingo responding favorably to your
> alternative, using task uid's to partition the tasks into CFS
> task-groups.
>
> Yeah, Ingo's preference for using uid's (or gid's ??) sounds right to
> me - a sustainable API.
>
> Wouldn't want to be adding a cpuset API for a single 2.6.N release.
>
> .... gid's -- why not?

Or process or process groups, or all of the above :-)

One thing to think on though, we cannot have per process,uid,gid,pgrp scheduling for one release only. So we'd have to manage interaction with process containers. It might be that a simple weight multiplication scheme is good enough:

weight = uid_weight * pgrp_weight * container_weight

Of course, if we'd only have a single level group scheduler (as was proposed IIRC) it'd have to create intersection sets (as there might be non trivial overlaps) based on these various weights and schedule these

resulting sets instead of the initial groupings.

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

---

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Peter Zijlstra on Wed, 11 Jul 2007 11:30:40 GMT
View Forum Message <> Reply to Message

On Wed, 2007-07-11 at 13:24 +0200, Peter Zijlstra wrote:
> On Wed, 2007-07-11 at 04:10 -0700, Paul Jackson wrote:
> > Srivatsa wrote:
> > > So Ingo was proposing we use cpuset as that user interface to manage
> > > task-groups. This will be only for 2.6.23.
> >
> > Good explanation - thanks.
> >
> > In short, the proposal was to use the task partition defined by cpusets
> > to define CFS task-groups, until the real process containers are
> > available.
> >
> > Or, I see in the next message, Ingo responding favorably to your
> > alternative, using task uid's to partition the tasks into CFS
> > task-groups.
> >
> > Yeah, Ingo's preference for using uid's (or gid's ??) sounds right to
> > me - a sustainable API.
> >
> > Wouldn't want to be adding a cpuset API for a single 2.6.N release.
> >
> > .... gid's -- why not?
>
>
> Or process or process groups, or all of the above :-)
>
> One thing to think on though, we cannot have per process,uid,gid,pgrp
> scheduling for one release only. So we'd have to manage interaction with
> process containers. It might be that a simple weight multiplication
> scheme is good enough:
>
> weight = uid_weight * pgrp_weight * container_weight
>
> Of course, if we'd only have a single level group scheduler (as was
> proposed IIRC) it'd have to create intersection sets (as there might be

> non trivial overlaps) based on these various weights and schedule these
> resulting sets instead of the initial groupings.

Lets illustrate with some ASCII art:

so we have this dual level weight grouping (uid, container)

uid: a a a a a b b b b b c c c c c
container: A A A A A A A B B B B B B B

set: 1 1 1 1 1 2 2 3 3 3 4 4 4 4 4

resulting in schedule sets 1,2,3,4

so that (for instance) weight_2 = weight_b * weight_A

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

---

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Srivatsa Vaddagiri on Wed, 11 Jul 2007 11:39:53 GMT
View Forum Message <> Reply to Message

On Wed, Jul 11, 2007 at 12:19:58PM +0200, Ingo Molnar wrote:
> * Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
>
> > The other alternative is to hook up group scheduler with user-id's
> > (again only for 2.6.23).
>
> could you just try this and send an as simple patch as possible? This is
> actually something that non-container people would be interested in as
> well.

Note that interfacing with container infrastructure doesn't preclude the
possibility of doing fair-user scheduling (that a normal university server or
desktop user would want). All that is needed is a daemon which listens for uid
change events from kernel (using process-event connector) and moves the task
(whose uid is changing) to an appropriate container for that user.
Primitive source for such a daemon is attached.

> (btw., if this goes into 2.6.23 then we cannot possibly turn it off in 2.6.24,

The fact that we will have two interface for group scheduler in 2.6.24
is what worries me a bit (one user-id based and other container based).
We would need some mechanism for admin to choose only one interface (and

not both together, otherwise the group definitions may conflict), which
doesn't sound very clean to me.

Ideally I would have liked to hook onto only container infrastructure
and let user-space decide grouping policy (whether user-id based or
something else).

Hmm ..would it help if I maintain a patch outside the mainline which turns on
fair-user scheduling in 2.6.23-rcX? Folks will have to apply that patch on
top of 2.6.23-rcX to use and test fair-user scheduling.

In 2.6.24, when container infrastructure goes in, people can get
fair-user scheduling off-the-shelf by simply starting the daemon attached
at bootup/initrd time.

Or would you rather prefer that I add user-id based interface
permanently and in 2.6.24 introduce a compile/run-time switch for admin
to select one of the two interfaces (user-id based or container-based)?

> so it must be sane - but per UID task groups are
> certainly sane, the only question is how to configure the per-UID weight
> after bootup.

Yeah ..the container based infrastructure allows for configuring such
things very easily using a fs-based interface. In the absence of that,
we either provide some /proc interface or settle for the non-configurable
default that you mention below.

> [the default after-bootup should be something along the
> lines of 'same weight for all users, a bit more for root'.]) This would
> make it possible for users to test that thing. (it would also help
> X-heavy workloads.)

--
Regards,
vatsa

```
/*
 * cpuctl_group_changer.c
 *
 * Used to change the group of running tasks to the correct
 * uid container.
 *
 * Copyright IBM Corporation, 2007
 * Author: Dhaval Giani <dhaval@linux.vnet.ibm.com>
 * Derived from test_cn_proc.c by Matt Helsley
 * Original copyright notice follows
```

```
 *
 * Copyright (C) Matt Helsley, IBM Corp. 2005
 * Derived from fcctl.c by Guillaume Thouvenin
 * Original copyright notice follows:
 *
 * Copyright (C) 2005 BULL SA.
 * Written by Guillaume Thouvenin <guillaume.thouvenin@bull.net>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <string.h>

#include <sys/socket.h>
#include <sys/types.h>

#include <sys/stat.h>
#include <sys/param.h>

#include <linux/connector.h>
#include <linux/netlink.h>
#include "linux/cn_proc.h"

#include <errno.h>

#include <signal.h>
#include <setjmp.h>

#define SEND_MESSAGE_LEN (NLMSG_LENGTH(sizeof(struct cn_msg) + \
        sizeof(enum proc_cn_mcast_op)))
#define RECV_MESSAGE_LEN (NLMSG_LENGTH(sizeof(struct cn_msg) + \
        sizeof(struct proc_event)))
```

```
#define SEND_MESSAGE_SIZE    (NLMSG_SPACE(SEND_MESSAGE_LEN))
#define RECV_MESSAGE_SIZE    (NLMSG_SPACE(RECV_MESSAGE_LEN))

#define max(x,y) ((y)<(x)?(x):(y))
#define min(x,y) ((y)>(x)?(x):(y))

#define BUFF_SIZE (max(max(SEND_MESSAGE_SIZE, RECV_MESSAGE_SIZE), 1024))
#define MIN_RECV_SIZE (min(SEND_MESSAGE_SIZE, RECV_MESSAGE_SIZE))

#define PROC_CN_MCAST_LISTEN (1)
#define PROC_CN_MCAST_IGNORE (2)

/*
 * SIGINT causes the program to exit gracefully
 * this could happen any time after the LISTEN message has
 * been sent
 */
#define INTR_SIG SIGINT


sigjmp_buf g_jmp;
char cpuctl_fs_path[MAXPATHLEN];

void handle_intr (int signum)
{
 siglongjmp(g_jmp, signum);
}

static inline void itos(int i, char* str)
{
 sprintf(str, "%d", i);
}

int set_notify_release(int val)
{
 FILE *f;

 f = fopen("notify_on_release", "r+");
 fprintf(f, "%d\n", val);
 fclose(f);
 return 0;
}

int add_task_pid(int pid)
{
 FILE *f;
```

```
 f = fopen("tasks", "a");
 fprintf(f, "%d\n", pid);
 fclose(f);
 return 0;
}

int set_value(char* file, char *str)
{
 FILE *f;

 f=fopen(file, "w");
 fprintf(f, "%s", str);
 fclose(f);
 return 0;
}

int change_group(int pid, int uid)
{
 char str[100];
 int ret;

 ret = chdir(cpuctl_fs_path);
 itos(uid, str);
 ret = mkdir(str, 0777);
 if (ret == -1) {
  /*
   * If the folder already exists, then it is alright. anything
   * else should be killed
   */
  if (errno != EEXIST) {
   perror("mkdir");
   return -1;
  }
 }
 ret = chdir(str);
 if (ret == -1) {
  /*Again, i am just quitting the program!*/
  perror("chdir");
  return -1;
 }
 /*If using cpusets set cpus and mems*
  *
  * set_value("cpus","0");
  * set_value("mems","0");
  */
 set_notify_release(1);
 add_task_pid(pid);
 return 0;
```

```
}

int handle_msg (struct cn_msg *cn_hdr)
{
 struct proc_event *ev;
 int ret;

 ev = (struct proc_event*)cn_hdr->data;

 switch(ev->what){
 case PROC_EVENT_UID:
  printf("UID Change happening\n");
  printf("UID = %d\tPID=%d\n", ev->event_data.id.e.euid,
      ev->event_data.id.process_pid);
  ret = change_group(ev->event_data.id.process_pid,
     ev->event_data.id.r.ruid);
  break;
 case PROC_EVENT_FORK:
 case PROC_EVENT_EXEC:
 case PROC_EVENT_EXIT:
 default:
  break;
 }
 return ret;
}
int main(int argc, char **argv)
{
 int sk_nl;
 int err;
 struct sockaddr_nl my_nla, kern_nla, from_nla;
 socklen_t from_nla_len;
 char buff[BUFF_SIZE];
 int rc = -1;
 struct nlmsghdr *nl_hdr;
 struct cn_msg *cn_hdr;
 enum proc_cn_mcast_op *mcop_msg;
 size_t recv_len = 0;
 FILE *f;

 if (argc == 1)
  strcpy(cpuctl_fs_path, "/dev/cpuctl");
 else
  strcpy(cpuctl_fs_path, argv[1]);
 chdir(cpuctl_fs_path);
 f = fopen("tasks", "r");
 if (f == NULL) {
  printf("Container not mounted at %s\n", cpuctl_fs_path);
  return -1;
```

```
}
fclose(f);
f = fopen("notify_on_release", "r");
if (f == NULL) {
 printf("Container not mounted at %s\n", cpuctl_fs_path);
 return -1;
}
fclose(f);
if (getuid() != 0) {
 printf("Only root can start/stop the fork connector\n");
 return 0;
}
/*
 * Create an endpoint for communication. Use the kernel user
 * interface device (PF_NETLINK) which is a datagram oriented
 * service (SOCK_DGRAM). The protocol used is the connector
 * protocol (NETLINK_CONNECTOR)
 */
sk_nl = socket(PF_NETLINK, SOCK_DGRAM, NETLINK_CONNECTOR);
if (sk_nl == -1) {
 printf("socket sk_nl error");
 return rc;
}
my_nla.nl_family = AF_NETLINK;
my_nla.nl_groups = CN_IDX_PROC;
my_nla.nl_pid = getpid();

kern_nla.nl_family = AF_NETLINK;
kern_nla.nl_groups = CN_IDX_PROC;
kern_nla.nl_pid = 1;

err = bind(sk_nl, (struct sockaddr *)&my_nla, sizeof(my_nla));
if (err == -1) {
 printf("binding sk_nl error");
 goto close_and_exit;
}
nl_hdr = (struct nlmsghdr *)buff;
cn_hdr = (struct cn_msg *)NLMSG_DATA(nl_hdr);
mcop_msg = (enum proc_cn_mcast_op*)&cn_hdr->data[0];
printf("sending proc connector: PROC_CN_MCAST_LISTEN... ");
memset(buff, 0, sizeof(buff));
*mcop_msg = PROC_CN_MCAST_LISTEN;
signal(INTR_SIG, handle_intr);
/* fill the netlink header */
nl_hdr->nlmsg_len = SEND_MESSAGE_LEN;
nl_hdr->nlmsg_type = NLMSG_DONE;
nl_hdr->nlmsg_flags = 0;
nl_hdr->nlmsg_seq = 0;
```

```
 nl_hdr->nlmsg_pid = getpid();
 /* fill the connector header */
 cn_hdr->id.idx = CN_IDX_PROC;
 cn_hdr->id.val = CN_VAL_PROC;
 cn_hdr->seq = 0;
 cn_hdr->ack = 0;
 cn_hdr->len = sizeof(enum proc_cn_mcast_op);
 if (send(sk_nl, nl_hdr, nl_hdr->nlmsg_len, 0) != nl_hdr->nlmsg_len) {
  printf("failed to send proc connector mcast ctl op!\n");
  goto close_and_exit;
 }
 printf("sent\n");
 for(memset(buff, 0, sizeof(buff)), from_nla_len = sizeof(from_nla);
   ; memset(buff, 0, sizeof(buff)), from_nla_len = sizeof(from_nla)) {
  struct nlmsghdr *nlh = (struct nlmsghdr*)buff;
  memcpy(&from_nla, &kern_nla, sizeof(from_nla));
  recv_len = recvfrom(sk_nl, buff, BUFF_SIZE, 0,
    (struct sockaddr*)&from_nla, &from_nla_len);
  if (recv_len < 1)
   continue;
  while (NLMSG_OK(nlh, recv_len)) {
   cn_hdr = NLMSG_DATA(nlh);
   if (nlh->nlmsg_type == NLMSG_NOOP)
    continue;
   if ((nlh->nlmsg_type == NLMSG_ERROR) ||
       (nlh->nlmsg_type == NLMSG_OVERRUN))
    break;
   if(handle_msg(cn_hdr)<0) {
    goto close_and_exit;
   }
   if (nlh->nlmsg_type == NLMSG_DONE)
    break;
   nlh = NLMSG_NEXT(nlh, recv_len);
  }
 }
close_and_exit:
 close(sk_nl);
 exit(rc);

 return 0;
}
```

File Attachments
1) cpuctld.c, downloaded 333 times

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Paul Jackson on Wed, 11 Jul 2007 11:42:44 GMT
View Forum Message <> Reply to Message

Srivatsa wrote:
> The fact that we will have two interface for group scheduler in 2.6.24
> is what worries me a bit (one user-id based and other container based).

Yeah.

One -could- take linear combinations, as Peter drew in his ascii art,
but would one -want- to do that?

I imagine some future time, when users of this wonder why the API is
more complicated than seems necessary, with two factors determining
task-groups where one seems sufficient, and the answer is "the other
factor, user-id's, is just there because we needed it as an interim
mechanism, and then had to keep it, to preserve ongoing compatibility.
That's not a very persuasive justification.

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

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

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Peter Zijlstra on Wed, 11 Jul 2007 12:06:33 GMT
View Forum Message <> Reply to Message

On Wed, 2007-07-11 at 04:42 -0700, Paul Jackson wrote:
> Srivatsa wrote:
> > The fact that we will have two interface for group scheduler in 2.6.24
> > is what worries me a bit (one user-id based and other container based).
>
> Yeah.
>
> One -could- take linear combinations, as Peter drew in his ascii art,
> but would one -want- to do that?

I'd very much like to have it, but that is just me. We could take a weight of 0 to mean disabling of that grouping and default to that. That way it would not complicate regular behaviour.

It could be implemented with a simple hashing scheme where sched_group_hash(tsk) and sched_group_cmp(tsk, group->some_task) could be used to identify a schedule group.

pseudo code:

```
u64 sched_group_hash(struct task_struct *tsk)
{
 u64 hash = 0;

 if (tsk->pid->weight)
  hash_add(&hash, tsk->pid);

 if (tsk->pgrp->weight)
  hash_add(&hash, tsk->pgrp);

 if (tsk->uid->weight)
  hash_add(&hash, tsk->uid);

 if (tsk->container->weight)
  hash_add(&hash, tsk->container);

 ...

 return hash;
}

s64 sched_group_cmp(struct task_struct *t1, struct task_struct *t2)
{
 s64 cmp;

 if (t1->pid->weight || t2->pid->weight) {
  cmp = t1->pid->weight - t2->pid->weight;
  if (cmp)
   return cmp;
 }

 ...

 return 0;
}

u64 sched_group_weight(struct task_struct *tsk)
```

```
{
 u64 weight = 1024; /* 1 fixed point 10 bits */

 if (tsk->pid->weight) {
  weight *= tsk->pid->weight;
  weight /= 1024;
 }

 ....

 return weight;
}
```

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

---

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Srivatsa Vaddagiri on Wed, 11 Jul 2007 12:30:48 GMT
View Forum Message <> Reply to Message

On Wed, Jul 11, 2007 at 05:09:53PM +0530, Srivatsa Vaddagiri wrote:
> > (btw., if this goes into 2.6.23 then we cannot possibly turn it off in 2.6.24,
>
> The fact that we will have two interface for group scheduler in 2.6.24
> is what worries me a bit (one user-id based and other container based).

I know breaking user-interface is a bad thing across releases. But in this particular case, it's probably ok (since fair-group scheduling is a brand new feature in Linux)?

If we have that option of breaking API between 2.6.23 and 2.6.24 for fair-group scheduler, then we are in a much more flexible position.

For 2.6.23, I can send a user-id based interface for fair-group scheduler (with some /proc interface to tune group nice value).

For 2.6.24, this user-id interface will be removed and we will instead switch to container based interface. Fair-user scheduling will continue to work, its just that users will have to use a daemon (sources sent in previous mail) to enable it on top of container-based interface.

Hmm..?

--
Regards,
vatsa

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

---

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Srivatsa Vaddagiri on Wed, 11 Jul 2007 13:14:05 GMT
View Forum Message <> Reply to Message

On Wed, Jul 11, 2007 at 01:30:40PM +0200, Peter Zijlstra wrote:
> > One thing to think on though, we cannot have per process,uid,gid,pgrp
> > scheduling for one release only. So we'd have to manage interaction with
> > process containers. It might be that a simple weight multiplication
> > scheme is good enough:
> >
> > weight = uid_weight * pgrp_weight * container_weight

We would need something like this to flatten hierarchy, so that for
example it is possible to do fair-container scheduling +
fair-user/process scheduling inside a container using a hierarchy depth of
just 1 (containers) that core scheduler understands. We discussed this a bit at
http://marc.info/?l=linux-kernel&m=118054481416140&w=2 and is very much
on my todo list to experiment with.

> > Of course, if we'd only have a single level group scheduler (as was
> > proposed IIRC) it'd have to create intersection sets (as there might be
> > non trivial overlaps) based on these various weights and schedule these
> > resulting sets instead of the initial groupings.
>
> Lets illustrate with some ASCII art:
>
> so we have this dual level weight grouping (uid, container)
>
> uid: a a a a a b b b b b c c c c c
> container: A A A A A A A B B B B B B B B
>
> set: 1 1 1 1 1 2 2 3 3 3 4 4 4 4 4
>
> resulting in schedule sets 1,2,3,4

Wouldn't it be simpler if admin created these sets as containers
directly? i.e:

uid: a a a a a b b b b b c c c c c
container: 1 1 1 1 1 2 2 3 3 3 4 4 4 4 4

That way scheduler will not have to "guess" such intersecting schedulable sets/groups. It seems much simpler to me this way.

Surely there is some policy which is driving some tasks of userid 'b' to be in container A and some to be in B. It should be trivial enough to hook onto that policy making script and create separate containers like above.

> so that (for instance) weight_2 = weight_b * weight_A

--
Regards,
vatsa

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Paul Jackson on Wed, 11 Jul 2007 17:03:31 GMT
View Forum Message <> Reply to Message

Peter wrote:
> I'd very much like to have it, but that is just me.

Why? [linear combinations of uid, container, pid, pgrp weighting]

You provide some implementation details and complications, but no motivation that I noticed.

Well ... a little motivation ... "just me", which would go a long way of your first name was Linus. For the rest of us ... ;).

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Peter Zijlstra on Wed, 11 Jul 2007 18:47:48 GMT
View Forum Message <> Reply to Message

On Wed, 2007-07-11 at 10:03 -0700, Paul Jackson wrote:
> Peter wrote:
> > I'd very much like to have it, but that is just me.
>
> Why? [linear combinations of uid, container, pid, pgrp weighting]

Good question, and I really have no other answer than that it seems usefull and not impossible (or even hard) to implement :-/

I'm not even that interested in using it, it just seems like a nice idea.

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Paul Menage on Wed, 11 Jul 2007 19:44:42 GMT
View Forum Message <> Reply to Message

On 7/10/07, Andrew Morton <akpm@linux-foundation.org> wrote:
>
> I'm inclined to take the cautious route here - I don't think people will be
> dying for the CFS thingy (which I didn't even know about?) in .23, and it's
> rather a lot of infrastructure to add for a CPU scheduler configurator

Selecting the relevant patches to give enough of the container framework to support a CFS container subsystem (slightly tweaked/updated versions of the base patch, procfs interface patch and tasks file interface patch) is about 1600 lines in kernel/container.c and another 200 in kernel/container.h, which is about 99% of the non-documentation changes.

So not tiny, but it's not very intrusive on the rest of the kernel, and would avoid having to introduce a temporary API based on uids.

Paul

Subject: Re: containers (was Re: -mm merge plans for 2.6.23)
Posted by Srivatsa Vaddagiri on Thu, 12 Jul 2007 05:39:35 GMT
View Forum Message <> Reply to Message

On Wed, Jul 11, 2007 at 12:44:42PM -0700, Paul Menage wrote:
> >I'm inclined to take the cautious route here - I don't think people will be
> >dying for the CFS thingy (which I didn't even know about?) in .23, and it's
> >rather a lot of infrastructure to add for a CPU scheduler configurator
>
> Selecting the relevant patches to give enough of the container
> framework to support a CFS container subsystem (slightly
> tweaked/updated versions of the base patch, procfs interface patch and
> tasks file interface patch) is about 1600 lines in kernel/container.c
> and another 200 in kernel/container.h, which is about 99% of the
> non-documentation changes.
>
> So not tiny, but it's not very intrusive on the rest of the kernel,
> and would avoid having to introduce a temporary API based on uids.

Yes that would be good. As long as the user-land interface for process containers doesn't change (much?) between 2.6.23 and later releases this should be a good workaround for us.

--
Regards,
vatsa

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