Subject: [RFC] libcg: design and plans
Posted by Dhaval Giani on Tue, 04 Mar 2008 15:23:41 GMT
View Forum Message <> Reply to Message

Hi,

We have been working on a library for control groups which would provide simple APIs for programmers to utilize from userspace and make use of control groups.

We are still designing the library and the APIs. I've attached the design (as of now) to get some feedback from the community whether we are heading in the correct direction and what else should be addressed.

We have a project on sourceforge.net at https://sourceforge.net/projects/libcg and the mailing list (cc'd here) can be found at https://lists.sourceforge.net/lists/listinfo/libcg-devel

Thanks,
--

libcg
1. Aims/Requirements
2. Design
3. APIs
4. Configuration Scheme

1. Aims/Requirements

1.1 What are Control Groups

Control Groups provide a mechanism for aggregating/partitioning sets of tasks, and all their future children, into hierarchical groups with specialized behaviour [1]. It makes use of a filesystem interface.

1.2 Aims of libcg

libcg aims to provide programmers easily usable APIs to use the control group file system. It should satisfy the following requirements

1.2.1. Provide a programmable interface for cgroups

This should allow applications to create cgroups using something like create_cgroup() as opposed to having to go the whole filesystem route.

1.2.2. Provide persistent configuration across reboots

Control Groups have a lifetime of only one boot cycle. The configuration

is lost at reboot. Userspace needs to handle this issue. This is handled by libcg

1.2.3. Provide a programmable interface for manipulating configurations

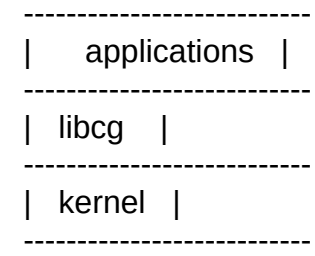
This should allow libcg to handle changing application requirements. For example, while gaming, you might want to reduce the cpu power of other groups whereas othertimes you would want greater CPU power for those groups.

2. Design

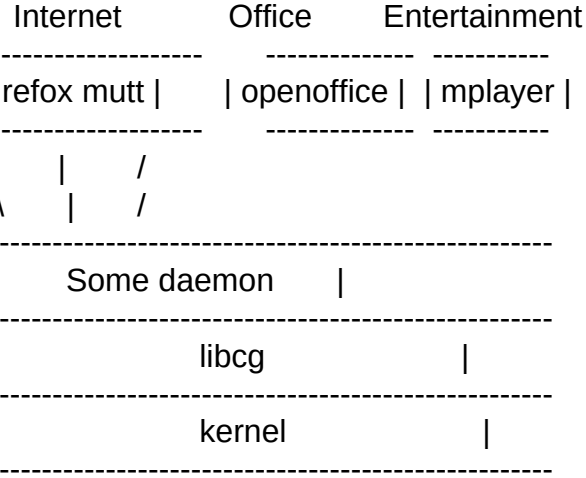
2.1 Architecture

2.1.1 Global overview

libcg will be consumed in the following fashion



A more detailed example would be as follows. Consider various applications running at the same time on a system. A typical system would be running a web browser, a mail client, a media player and office software. libcg could be used to group these applications into various groups and give them various resources. A possible example would be three groups, Internet, Entertainment and Office. A daemon could attach tasks to these groups according to some rules and the adminstrator can control the resources attached to each group via the configuration manager.i

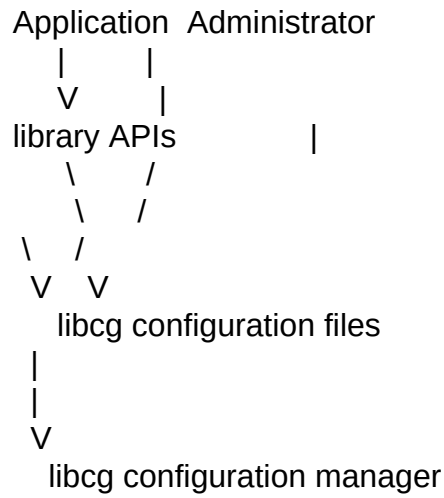


2.1.2

libcg will consist of two main parts. The configuration manager and the library.

The configuration manager will used to maintain the configurations, to load and unload the configurations, to set the bootup configurations and so on. This is similar to the network configuration. A configuration file is used to setup the networking at bootup. Similarly a configuration file will be used to setup the default control groups (and maybe the top level control groups) at bootup.

The administrator can directly access the configuration files, and applications can access it through the library. The configuration manager is used to provide the persistence.

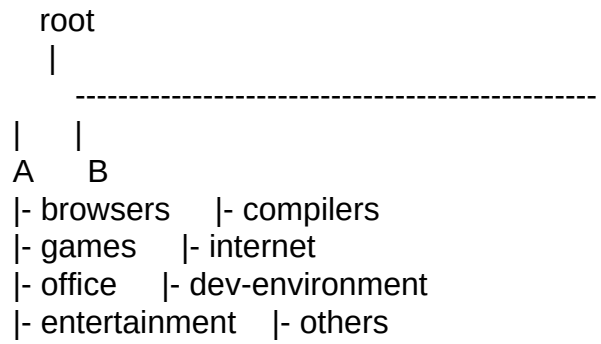


The configuration manager has to provide isolation between various users of libcg. That is, if two different users A and B are making use of libcg, then the configuration manager has to ensure that user A does not affect user B's settings/configurations.

The top level limits and permissions for A and B are to be provided by the administrator. The permissions are filesystem permissions as cgroup is filesystem based.

With this architecure in mind, we expect two levels of configuration files. One would be the global configuration which the administrator would control and setup the groups, and a local configuration which the group owner will control.

A simple example could be that the administrator could split the top level according to uid, and then each user could control the resources available to him and group those applications accordingly.



In this example, we have an example cgroup filesystem configuration.

The administrator decides the resources available to "A" and "B". Both "A" and "B" have followed grouping according to their usage. They decide the resources availble to their groups (which is dependent of the resources alloted to them by the adminstrator).

libcg will be written mainly in C with lex and yacc for parsing the configuration files.

3. APIs

The APIs are envisaged to be of two main types

3.1. Manipulating Control Groups

3.1.1. Create Control Group: This API is proposed to create control groups. It should take care of the following scenarios

3.1.1.1 Create non persistent control groups: These groups should exist for just duration of this run. They should not stick across different sessions.

3.1.1.2 Create persistent control groups: These groups should stick across different sessions.

3.1.2 Delete Control Group: This API is proposed to delete control groups. It would have the same scenarios as expected for Create Control Group.

3.1.2 Modify Control Group: This API proposed to modify an already existing group's control files. It too should handle the persistence issue as like Create Control Group does.

More details about configuration are available in sections 2 and 4.

3.2. Manipulating Configurations

3.2.1. Generate Configuration File: If a cgroup filesystem hierarchy already exists,

it should be possible to generate a configuration file which can create it. This is proposed to be provided by this API.

3.2.2. Change Configuration File: If one configuration is currently loaded in memory, it is possible for it to be replaced with the new file. This API proposes to implement that.

3.2.3. Manipulate Configuration File: This API proposes to allow the configuration file to be modified.

We should also plan on taking care of statistics once its available in mainline.

4. Configuration Scheme

There are multiple configuration levels. The basic wlm.conf file will provide the mount points and the controller details. This can only be manipulated by the adminstrator. No APIs will be provided to modify this file.

There will be group specific configuration files as well. The exact details of the same still need to be worked out.

4.1. Sample configuration files

4.1.1. Sample wlm.conf

```
#
# controller file
#

group ca1 {
 perm {
  task {
   uid = balbir;
   gid = cgroup;
  }
  admin {
   uid = root;
   gid = cgroup;
  }
 }

 cpu {
  cpu.shares = 500;
 }
}

mount {
 cpu = /container;
```

}

This is an example of a top level group. The mount{} block is used to provide the mount point of the various controllers. For eg, the cpu controller is mounted at /container. Next we have the group ca1. This is the top level group and its permissions are given by the uid and gid fields for tasks and admin. The next is the individual controller block. For the mount point of cpu, the cpu.shares value is provided. Thus the above file can be represented as the following script

```
mkdir /container
mount -t cgroup -o cpu none /container
mkdir /container/ca1
/bin/echo 500 > /container/ca1/cpu.shares
chown -R root /container/ca1
chgrp -R cgroup /container/ca1
chown balbir /container/ca1/tasks
chgrp cgroup /container/ca1/tasks
```

5. References

1. Documentation/cgroups.txt in kernel sources.

--
regards,
Dhaval

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by xpl on Tue, 04 Mar 2008 17:13:51 GMT
View Forum Message <> Reply to Message

Hi,

I was wonder if creating such library makes any sense at all,
considering the nature of cgroups, the way they work and their possible
application?
It seems to me that any attempt to create a 'simple' API would actualy
result in something that will be much harder to use that just making raw
mkdir/open/read/write/close operations. Another thing is suggested
config for this lib would be more appropriate for a daemon rather than a
library.
In general - cgroup is a very flexible subsystem that can be used in a
wide variety of ways and modes and trying to create a universal simple
API would more likely result in something hard to manage and work with.
The idea of having multiple container managers (applications that use

libcg) creates a lots of questions and possible issues. Containers are supposed provide a flexible resource management and task grouping ability, which somewhat implies that there cannot be two different resource managers per node without one knowing well the desires/goals/methods of the other and vice versa. And should there be only one manager per node - probably it will be easier for it to use cgroup subsystem directly rather than using a wrapper library?

Regards,
Peter Litov.

Dhaval Giani ??????:
> Hi,
>
> We have been working on a library for control groups which would provide
> simple APIs for programmers to utilize from userspace and make use of
> control groups.
>
> We are still designing the library and the APIs. I've attached the
> design (as of now) to get some feedback from the community whether we
> are heading in the correct direction and what else should be addressed.
>
> We have a project on sourceforge.net at
> https://sourceforge.net/projects/libcg and the mailing list (cc'd here)
> can be found at https://lists.sourceforge.net/lists/listinfo/libcg-devel
>
> Thanks,
> --
>
> libcg
> 1. Aims/Requirements
> 2. Design
> 3. APIs
> 4. Configuration Scheme
>
> 1. Aims/Requirements
>
> 1.1 What are Control Groups
>
> Control Groups provide a mechanism for aggregating/partitioning sets of
> tasks, and all their future children, into hierarchical groups with
> specialized behaviour [1]. It makes use of a filesystem interface.
>
> 1.2 Aims of libcg
>
> libcg aims to provide programmers easily usable APIs to use the control
> group file system. It should satisfy the following requirements
>

> 1.2.1. Provide a programmable interface for cgroups
>
> This should allow applications to create cgroups using something like
> create_cgroup() as opposed to having to go the whole filesystem route.
>
> 1.2.2. Provide persistent configuration across reboots
>
> Control Groups have a lifetime of only one boot cycle. The configuration
> is lost at reboot. Userspace needs to handle this issue. This is handled
> by libcg
>
> 1.2.3. Provide a programmable interface for manipulating configurations
>
> This should allow libcg to handle changing application requirements. For
> example, while gaming, you might want to reduce the cpu power of other
> groups whereas othertimes you would want greater CPU power for those
> groups.
>
> 2. Design
>
> 2.1 Architecture
>
> 2.1.1 Global overview
>
> libcg will be consumed in the following fashion
>
```
>    ---------------------------
>    |      applications   |
>    ---------------------------
>    |   libcg    |
>    ---------------------------
>    |   kernel   |
>    ---------------------------
```
>
> A more detailed example would be as follows. Consider various applications
> running at the same time on a system. A typical system would be running
> a web browser, a mail client, a media player and office software. libcg
> could be used to group these applications into various groups and give
> them various resources. A possible example would be three groups, Internet,
> Entertainment and Office. A daemon could attach tasks to these groups according
> to some rules and the adminstrator can control the resources attached to each
> group via the configuration manager.i
>
```
>      Internet           Office        Entertainment
>  ----------------------       --------------  -----------
>  | firefox mutt |       | openoffice |  | mplayer |
>  ----------------------       --------------  -----------
>    \       |        /
```

```
>    \      |      /
>   ----------------------------------------------------
>   |        Some daemon       |
>   ----------------------------------------------------
>   |                    libcg                    |
>   ----------------------------------------------------
>   |                    kernel                    |
>   ----------------------------------------------------
>
> 2.1.2
>
> libcg will consist of two main parts. The configuration manager and the
> library.
>
> The configuration manager will used to maintain the configurations, to load
> and unload the configurations, to set the bootup configurations and so on.
> This is similar to the network configuration. A configuration file is used
> to setup the networking at bootup. Similarly a configuration file will be
> used to setup the default control groups (and maybe the top level control
> groups) at bootup.
>
> The administrator can directly access the configuration files, and
> applications can access it through the library. The configuration manager
> is used to provide the persistence.
>
>  Application  Administrator
>       |        |
>       V        |
>  library APIs              |
>        \       /
>         \     /
>   \    /
>    V    V
>       libcg configuration files
>    |
>    |
>    V
>      libcg configuration manager
>
> The configuration manager has to provide isolation between various users of
> libcg. That is, if two different users A and B are making use of libcg, then
> the configuration manager has to ensure that user A does not affect user B's
> settings/configurations.
>
> The top level limits and permissions for A and B are to be provided by the
> administrator. The permissions are filesystem permissions as cgroup is
> filesystem based.
>
```

> With this architecure in mind, we expect two levels of configuration files.
> One would be the global configuration which the administrator would control
> and setup the groups, and a local configuration which the group owner will
> control.
>
> A simple example could be that the administrator could split the top level
> according to uid, and then each user could control the resources available
> to him and group those applications accordingly.
>
> root
> |
> ----------
> | |
> A B
> |- browsers |- compilers
> |- games |- internet
> |- office |- dev-environment
> |- entertainment |- others
>
> In this example, we have an example cgroup filesystem configuration.
>
> The administrator decides the resources available to "A" and "B". Both "A" and
> "B" have followed grouping according to their usage. They decide the resources
> availble to their groups (which is dependent of the resources alloted to them
> by the adminstrator).
>
> libcg will be written mainly in C with lex and yacc for parsing the configuration
> files.
>
> 3. APIs
>
> The APIs are envisaged to be of two main types
>
> 3.1. Manipulating Control Groups
>
> 3.1.1. Create Control Group: This API is proposed to create control groups.
> It should take care of the following scenarios
>
> 3.1.1.1 Create non persistent control groups: These groups should exist
> for just duration of this run. They should not stick across different sessions.
>
> 3.1.1.2 Create persistent control groups: These groups should stick across
> different sessions.
>
> 3.1.2 Delete Control Group: This API is proposed to delete control groups.
> It would have the same scenarios as expected for Create Control Group.
>
> 3.1.2 Modify Control Group: This API proposed to modify an already

> existing group's control files. It too should handle the persistence issue
> as like Create Control Group does.
>
> More details about configuration are available in sections 2 and 4.
>
> 3.2. Manipulating Configurations
>
> 3.2.1. Generate Configuration File: If a cgroup filesystem hierarchy already exists,
> it should be possible to generate a configuration file which can create it. This
> is proposed to be provided by this API.
>
> 3.2.2. Change Configuration File: If one configuration is currently loaded in
> memory, it is possible for it to be replaced with the new file. This API proposes
> to implement that.
>
> 3.2.3. Manipulate Configuration File: This API proposes to allow the configuration
> file to be modified.
>
> We should also plan on taking care of statistics once its available in mainline.
>
> 4. Configuration Scheme
>
> There are multiple configuration levels. The basic wlm.conf file will provide
> the mount points and the controller details. This can only be manipulated by
> the adminstrator. No APIs will be provided to modify this file.
>
> There will be group specific configuration files as well. The exact details
> of the same still need to be worked out.
>
> 4.1. Sample configuration files
>
> 4.1.1. Sample wlm.conf
>
> #
> # controller file
> #
>
> group ca1 {
> perm {
>   task {
>   uid = balbir;
>   gid = cgroup;
>   }
>   admin {
>   uid = root;
>   gid = cgroup;
>   }
> }

>
> cpu {
>   cpu.shares = 500;
> }
> }
>
> mount {
> cpu = /container;
> }
>
> This is an example of a top level group. The mount{} block is used to provide
> the mount point of the various controllers. For eg, the cpu controller is
> mounted at /container. Next we have the group ca1. This is the top level group
> and its permissions are given by the uid and gid fields for tasks and admin. The
> next is the individual controller block. For the mount point of cpu, the cpu.shares
> value is provided. Thus the above file can be represented as the following script
>
> mkdir /container
> mount -t cgroup -o cpu none /container
> mkdir /container/ca1
> /bin/echo 500 > /container/ca1/cpu.shares
> chown -R root /container/ca1
> chgrp -R cgroup /container/ca1
> chown balbir /container/ca1/tasks
> chgrp cgroup /container/ca1/tasks
>
> 5. References
>
> 1. Documentation/cgroups.txt in kernel sources.
>

_____

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by Dave Hansen on Tue, 04 Mar 2008 18:05:12 GMT
View Forum Message <> Reply to Message

On Tue, 2008-03-04 at 20:53 +0530, Dhaval Giani wrote:
> We have been working on a library for control groups which would provide
> simple APIs for programmers to utilize from userspace and make use of
> control groups.

What kinds of things are lacking the the cgroup interface that you want
to make up for in this library?

-- Dave

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by Balbir Singh on Wed, 05 Mar 2008 04:48:50 GMT
View Forum Message <> Reply to Message

Xpl++ wrote:
> Hi,
>
> I was wonder if creating such library makes any sense at all,
> considering the nature of cgroups, the way they work and their possible
> application?
> It seems to me that any attempt to create a 'simple' API would actualy
> result in something that will be much harder to use that just making raw
> mkdir/open/read/write/close operations. Another thing is suggested
> config for this lib would be more appropriate for a daemon rather than a
> library.

The configuration file is important, since it allows us two levels of control.
At one level the system administrator and at the other level applications. Each
application can maintain it's own hierarchy across reboots.

We thought of a daemon, but there were several overheads and disadvantages. For
one, we needed an IPC mechanism to communicate every client request to the
daemon, the client being the application. The daemon also becomes the single
point of failure for all applications. Moreover, we want to provide the ability
to programmatically update the configuration. A daemon, if desired can be built
on top of the library we propose.

> In general - cgroup is a very flexible subsystem that can be used in a
> wide variety of ways and modes and trying to create a universal simple
> API would more likely result in something hard to manage and work with.

I agree that cgroups is flexible, but why do you think abstracting common tasks
amongst applications will be hard to manage and work with? We want to provide
API that will allow us to fill in parameters and say -- go create this group or
delete this group.

> The idea of having multiple container managers (applications that use
> libcg) creates a lots of questions and possible issues. Containers are
> supposed provide a flexible resource management and task grouping

> ability, which somewhat implies that there cannot be two different
> resource managers per node without one knowing well the
> desires/goals/methods of the other and vice versa.

We don't assume that there cannot be two different resource managers per node.
We don't enforce any policy, we just allow for easy creation and manipulation of
control groups hierarchially.

And should there be
> only one manager per node - probably it will be easier for it to use
> cgroup subsystem directly rather than using a wrapper library?
>

Could you please elaborate, why is it probably easier?

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by Dhaval Giani on Wed, 05 Mar 2008 05:26:31 GMT
View Forum Message <> Reply to Message

Hi Peter,

Thanks for your comments.

On Tue, Mar 04, 2008 at 07:15:51PM +0200, Xpl++ wrote:
> Hi,
>
> I was wonder if creating such library makes any sense at all, considering
> the nature of cgroups, the way they work and their possible application?
> It seems to me that any attempt to create a 'simple' API would actualy
> result in something that will be much harder to use that just making raw
> mkdir/open/read/write/close operations.

These simple APIs are nothing but raw mkdir/open/read/write/close
operations.

> Another thing is suggested config
> for this lib would be more appropriate for a daemon rather than a library.

> In general - cgroup is a very flexible subsystem that can be used in a wide
> variety of ways and modes and trying to create a universal simple API would
> more likely result in something hard to manage and work with.
> The idea of having multiple container managers (applications that use
> libcg) creates a lots of questions and possible issues. Containers are
> supposed provide a flexible resource management and task grouping ability,
> which somewhat implies that there cannot be two different resource managers
> per node without one knowing well the desires/goals/methods of the other
> and vice versa. And should there be only one manager per node - probably it
> will be easier for it to use cgroup subsystem directly rather than using a
> wrapper library?

I disagree. Allowing multiple resource managers allows more flexibility.
One thing the configuration subsystem aims to do is to allow permissions
to the groups. With this happening, a resource manager does not need to
about the existence of other groups, it can control only the resources
allotted to it. And since the top level is controlled by the
administrator, he is aware of the groups and their needs. (I've given an
example of such a scenario in the document as well).

Hope this helps answer your question.

Thanks,
--
regards,
Dhaval

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by Paul Menage on Wed, 05 Mar 2008 06:15:20 GMT
View Forum Message <> Reply to Message

Hi Dhaval,

On Tue, Mar 4, 2008 at 7:23 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
> Hi,
>
> We have been working on a library for control groups which would provide
> simple APIs for programmers to utilize from userspace and make use of
> control groups.
>
> We are still designing the library and the APIs. I've attached the
> design (as of now) to get some feedback from the community whether we
> are heading in the correct direction and what else should be addressed.

There are a few things that it would be nice to include in such a
library, if you're going to develop one:

- the ability to create abstract groups of processes, and resource
groups, and have the ability to tie these together arbitrarily. E.g
you might create abstract groups A, B and C, and be able to say that A
and B share memory with each other but not with C, and all three
groups are isolated from each other for CPU. Then libcg would mount
different resource types in different cgroup hierarchies (you would
probably tell it ahead of time which combinations of sharing you would
want, in order that it could minimize the number of mounted
hierarchies). When you tell libcg to move a process into abstract
group A, it would move it into the appropriate resource group in each
hierarchy.

- an interface for gathering usage stats from cgroups.

- support for dynamically migrating processes between groups based on
process connector events (i.e. a finished version of the daemon that
you were working on last year)

Paul

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

---

Subject: Re: Re: [RFC] libcg: design and plans
Posted by den on Wed, 05 Mar 2008 07:17:47 GMT
View Forum Message <> Reply to Message

On Tue, 2008-03-04 at 22:15 -0800, Paul Menage wrote:
> Hi Dhaval,
>
> On Tue, Mar 4, 2008 at 7:23 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
> > Hi,
> >
> > We have been working on a library for control groups which would provide
> > simple APIs for programmers to utilize from userspace and make use of
> > control groups.
> >
> > We are still designing the library and the APIs. I've attached the
> > design (as of now) to get some feedback from the community whether we
> > are heading in the correct direction and what else should be addressed.
>
> There are a few things that it would be nice to include in such a

> library, if you're going to develop one:
>
> - the ability to create abstract groups of processes, and resource
> groups, and have the ability to tie these together arbitrarily. E.g
> you might create abstract groups A, B and C, and be able to say that A
> and B share memory with each other but not with C, and all three
> groups are isolated from each other for CPU. Then libcg would mount
> different resource types in different cgroup hierarchies (you would
> probably tell it ahead of time which combinations of sharing you would
> want, in order that it could minimize the number of mounted
> hierarchies). When you tell libcg to move a process into abstract
> group A, it would move it into the appropriate resource group in each
> hierarchy.

There is one more important thing. In addition to the processes you must
unite or provide a way to unite other objects like sockets. This is
needed to create a group-based socket buffer management.

The mapping between socket and a process does not exists right now and,
we can have (virtually), sockets from from different namespaces in one
process.

Regards,
Den

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by Dhaval Giani on Wed, 05 Mar 2008 10:33:43 GMT
View Forum Message <> Reply to Message

On Tue, Mar 04, 2008 at 10:15:20PM -0800, Paul Menage wrote:
> Hi Dhaval,
>
> On Tue, Mar 4, 2008 at 7:23 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
> > Hi,
> >
> > We have been working on a library for control groups which would provide
> > simple APIs for programmers to utilize from userspace and make use of
> > control groups.
> >
> > We are still designing the library and the APIs. I've attached the
> > design (as of now) to get some feedback from the community whether we
> > are heading in the correct direction and what else should be addressed.

>
> There are a few things that it would be nice to include in such a
> library, if you're going to develop one:
>
> - the ability to create abstract groups of processes, and resource
> groups, and have the ability to tie these together arbitrarily. E.g
> you might create abstract groups A, B and C, and be able to say that A
> and B share memory with each other but not with C, and all three
> groups are isolated from each other for CPU. Then libcg would mount
> different resource types in different cgroup hierarchies (you would
> probably tell it ahead of time which combinations of sharing you would
> want, in order that it could minimize the number of mounted
> hierarchies). When you tell libcg to move a process into abstract
> group A, it would move it into the appropriate resource group in each
> hierarchy.
>

I am not very clear about what you are asking for here, so let me try to rephrase it, and if I have understood it correctly, we can move further ahead from there.

So there are two different points, /mem and /cpu. /mem has A and C and /cpu has A, B and C. A and B of /cpu correspond to A of /mem and the C's are the same. With this is mind, if I say a task should move to B in /cpu, it should also move to A in /mem?

> - an interface for gathering usage stats from cgroups.
>

Yes, that is a todo. We should get around to it as the functionality gets implemented in kernel.

> - support for dynamically migrating processes between groups based on
> process connector events (i.e. a finished version of the daemon that
> you were working on last year)
>

libcg is at a lower level than this. The dynamic migration of processes can be based on top of libcg, and exploit it (and be more powerful than the daemon I posted last year) It would be able to utilize the configuration and other capabilities of libcg.

Thanks,
--
regards,
Dhaval

_____
Containers mailing list

Subject: Re: [RFC] libcg: design and plans
Posted by Paul Menage on Wed, 05 Mar 2008 10:41:41 GMT
View Forum Message <> Reply to Message

On Wed, Mar 5, 2008 at 2:33 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
>
> So there are two different points, /mem and /cpu. /mem has A and C and
> /cpu has A, B and C. A and B of /cpu correspond to A of /mem and the C's
> are the same. With this is mind, if I say a task should move to B in
> /cpu, it should also move to A in /mem?
>

Maybe clearer to say that /mem has two cgroups, AB and C. The
abstraction provided by libcg would be of three groups, A, B and C.
Asking libcg to move a process to abstract group B would result it
moving to /mem/AB and /cpu/B

Paul

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

Subject: Re: [RFC] libcg: design and plans
Posted by Dhaval Giani on Wed, 05 Mar 2008 11:07:31 GMT
View Forum Message <> Reply to Message

On Wed, Mar 05, 2008 at 02:41:41AM -0800, Paul Menage wrote:
> On Wed, Mar 5, 2008 at 2:33 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
> >
> > So there are two different points, /mem and /cpu. /mem has A and C and
> > /cpu has A, B and C. A and B of /cpu correspond to A of /mem and the C's
> > are the same. With this is mind, if I say a task should move to B in
> > /cpu, it should also move to A in /mem?
> >
>
> Maybe clearer to say that /mem has two cgroups, AB and C. The
> abstraction provided by libcg would be of three groups, A, B and C.
> Asking libcg to move a process to abstract group B would result it
> moving to /mem/AB and /cpu/B
>

OK. Hmm, I've not really thought about it. At first thought, it should
not be very difficult. Only thing I am not sure is the arbitrary
grouping of the groups (ok, a bit confusing). If that information is
maintained somewhere, it should be pretty straightforward. (Only thing
is that I am not sure how it will be done, and where the grouping
information should be stored. configuration looks like the logical
place, but I am not sure)

Thanks,
--
regards,
Dhaval

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

---

Subject: Re: Re: [RFC] libcg: design and plans
Posted by Balbir Singh on Wed, 05 Mar 2008 11:48:18 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> On Tue, 2008-03-04 at 22:15 -0800, Paul Menage wrote:
>> Hi Dhaval,
>>
>> On Tue, Mar 4, 2008 at 7:23 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
>>> Hi,
>>>
>>> We have been working on a library for control groups which would provide
>>> simple APIs for programmers to utilize from userspace and make use of
>>> control groups.
>>>
>>> We are still designing the library and the APIs. I've attached the
>>> design (as of now) to get some feedback from the community whether we
>>> are heading in the correct direction and what else should be addressed.
>> There are a few things that it would be nice to include in such a
>> library, if you're going to develop one:
>>
>> - the ability to create abstract groups of processes, and resource
>> groups, and have the ability to tie these together arbitrarily. E.g
>> you might create abstract groups A, B and C, and be able to say that A
>> and B share memory with each other but not with C, and all three
>> groups are isolated from each other for CPU. Then libcg would mount
>> different resource types in different cgroup hierarchies (you would
>> probably tell it ahead of time which combinations of sharing you would
>> want, in order that it could minimize the number of mounted
>> hierarchies). When you tell libcg to move a process into abstract

>> group A, it would move it into the appropriate resource group in each
>> hierarchy.
>
> There is one more important thing. In addition to the processes you must
> unite or provide a way to unite other objects like sockets. This is
> needed to create a group-based socket buffer management.
>
> The mapping between socket and a process does not exists right now and,
> we can have (virtually), sockets from from different namespaces in one
> process.
>

Not sure how any of this is related to the library design we are discussing.
Your talking about writing a controller that groups based on sockets, that is a
totally different thing.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by Paul Menage on Wed, 05 Mar 2008 11:51:01 GMT
View Forum Message <> Reply to Message

On Wed, Mar 5, 2008 at 3:07 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
>
> OK. Hmm, I've not really thought about it. At first thought, it should
> not be very difficult. Only thing I am not sure is the arbitrary
> grouping of the groups (ok, a bit confusing).

I suspect that the main form of composite grouping is going to be
between parents and children. E.g. you might want to say things like:

create_group(A, memory=1G, cpu=100)
create_group(B, parent=A, memory=inherit, cpu=20)
create_group(C, parent=A, memory=inherit, cpu=30)

i.e. both B and C inherit/share their memory limit from their parent,
but have their own CPU groups (child groups of their parent?)

So this would result in a single group A in the memory hierarchy and a

top-level group A and child groups B and C in the cpu hierarchy. libcg would abstract away the fact that when you moved a process into an abstract group, it actually had to be moved into multiple real groups.

I think this kind of sharing is fairly easy to specify. Now, there's no reason that it shouldn't support more complex group sharing as well, but that might require the user to use lower-level operations, such as creating resource groups in particular hierarchies, and associating abstract groups with those resource groups.

The model above (children sharing resource groups with their parents for some resources) is actually something that I figured could be supported relatively straightforwardly in the kernel - essentially:

- each subsystem "foo" would have a foo.inherit file provided by cgroups in each group directory

- setting the foo.inherit flag (i.e. writing 1 to it) would cause tasks in that cgroup to share the "foo" subsystem state with the parent cgroup

- from the subsystem's point of view, it would only need to worry about its own foo_cgroup objects and which task was associated with each object; the subsystem wouldn't need to care about which tasks were part of each cgroup, and which cgroups were sharing state; that would all be taken care of by the cgroup framework

I'd sketched out a fairly nice design for how it would all work in my head when I realised that it could actually all be done via multiple hierarchies in userspace with something like the libcg operations I suggested above.

Paul

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by xpl on Wed, 05 Mar 2008 11:56:23 GMT
View Forum Message <> Reply to Message

Hi Dhaval,

Dhaval Giani ??????:
>> I was wonder if creating such library makes any sense at all, considering
>> the nature of cgroups, the way they work and their possible application?

>> It seems to me that any attempt to create a 'simple' API would actualy
>> result in something that will be much harder to use that just making raw
>> mkdir/open/read/write/close operations.
>>
>
> These simple APIs are nothing but raw mkdir/open/read/write/close
> operations.
>
>
Then why do you have to make the api? Everybody knows how to user these
syscalls :)
If its only this -> why use it? If it's not -> it aint simple anymore?
>> Another thing is suggested config
>> for this lib would be more appropriate for a daemon rather than a library.
>> In general - cgroup is a very flexible subsystem that can be used in a wide
>> variety of ways and modes and trying to create a universal simple API would
>> more likely result in something hard to manage and work with.
>> The idea of having multiple container managers (applications that use
>> libcg) creates a lots of questions and possible issues. Containers are
>> supposed provide a flexible resource management and task grouping ability,
>> which somewhat implies that there cannot be two different resource managers
>> per node without one knowing well the desires/goals/methods of the other
>> and vice versa. And should there be only one manager per node - probably it
>> will be easier for it to use cgroup subsystem directly rather than using a
>> wrapper library?
>>
>
> I disagree. Allowing multiple resource managers allows more flexibility.
> One thing the configuration subsystem aims to do is to allow permissions
> to the groups. With this happening, a resource manager does not need to
> about the existence of other groups, it can control only the resources
> allotted to it. And since the top level is controlled by the
> administrator, he is aware of the groups and their needs. (I've given an
> example of such a scenario in the document as well).
>
Imagine having a shared/joint household savings account with your wife,
and taking money from it without your wife knowing and vice versa ..
then at some point when you thought that you have some $5K to buy the
new super duper laptop you dreamt about your entire life - surprise - no
enough resources :)
This is somewhat the equivalent of multiple independent resourse
managers :) It won't end well :)
Should they be expected to be adequate in doing their job, they cannot
be independent since they manage a shared resource.

Regards,
Peter.

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

Subject: Re: [RFC] libcg: design and plans
Posted by Paul Menage on Wed, 05 Mar 2008 12:01:55 GMT
View Forum Message <> Reply to Message

On Tue, Mar 4, 2008 at 7:23 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
>
> libcg will be written mainly in C with lex and yacc for parsing the configuration
> files.
>

One suggestion - for configuration management tools like this I'd opt
for C++ or Python over C, in order to be able to use STL (or the
python libraries). Object-Orientation may be useful or maybe not, but
classes like string, vector, hash_map, etc, make life so much easier.

Paul

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

Subject: Re: [RFC] libcg: design and plans
Posted by Balbir Singh on Wed, 05 Mar 2008 12:27:42 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> On Tue, Mar 4, 2008 at 7:23 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
>> libcg will be written mainly in C with lex and yacc for parsing the configuration
>> files.
>>
>
> One suggestion - for configuration management tools like this I'd opt
> for C++ or Python over C, in order to be able to use STL (or the
> python libraries). Object-Orientation may be useful or maybe not, but
> classes like string, vector, hash_map, etc, make life so much easier.
>
> Paul

Object-orientation is not very useful in this case. STIL is useful, if we don't
find equivalent functionality, we might consider using C++ or the C++ compiler
and libraries and export "C" interfaces. But for now, our plan is to use the C

compiler

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by Balbir Singh on Wed, 05 Mar 2008 14:24:56 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> On Wed, Mar 5, 2008 at 3:07 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
>> OK. Hmm, I've not really thought about it. At first thought, it should
>> not be very difficult. Only thing I am not sure is the arbitrary
>> grouping of the groups (ok, a bit confusing).
>
> I suspect that the main form of composite grouping is going to be
> between parents and children. E.g. you might want to say things like:
>
> create_group(A, memory=1G, cpu=100)
> create_group(B, parent=A, memory=inherit, cpu=20)
> create_group(C, parent=A, memory=inherit, cpu=30)
>
> i.e. both B and C inherit/share their memory limit from their parent,
> but have their own CPU groups (child groups of their parent?)
>

No, we don't plan on doing that. What we plan on doing is

1. Specify the mount point for each controller
2. In the create group API, specify the name of the group and the various
parameters.

If for example CPU is mounted at /cpu and Memory at /mem

Then a specification for creation of group A would be of the form

create_group(A, cpu=100, memory=100M)

Then,

/cpu/A has shares set to 100 and /mem/A has memory.limit set to 100M

If you want to create subgroups under A, you specify

create_group(A/B, memory=200M, cpu=50)

That would create /cpu/A/B and /mem/A/B

Please note that memory and CPU hierarchy needs work in the kernel. The shares and hierarchy support is pending. We need to make the res_counters infrastructure aware of hierarchies.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by Dhaval Giani on Wed, 05 Mar 2008 15:53:10 GMT
View Forum Message <> Reply to Message

On Wed, Mar 05, 2008 at 01:56:23PM +0200, Xpl++ wrote:
> Hi Dhaval,
>
> Dhaval Giani ??????:
>>> I was wonder if creating such library makes any sense at all, considering
>>> the nature of cgroups, the way they work and their possible application?
>>> It seems to me that any attempt to create a 'simple' API would actualy
>>> result in something that will be much harder to use that just making raw
>>> mkdir/open/read/write/close operations.
>>>
>>
>> These simple APIs are nothing but raw mkdir/open/read/write/close
>> operations.
>>
>>
> Then why do you have to make the api? Everybody knows how to user these
> syscalls :)
> If its only this -> why use it? If it's not -> it aint simple anymore?

Well, that's just a small subset of the APIs. There are a lot of other
things that libcg does.

>>> Another thing is suggested config for this lib would be more appropriate
>>> for a daemon rather than a library.
>>> In general - cgroup is a very flexible subsystem that can be used in a
>>> wide variety of ways and modes and trying to create a universal simple
>>> API would more likely result in something hard to manage and work with.
>>> The idea of having multiple container managers (applications that use
>>> libcg) creates a lots of questions and possible issues. Containers are
>>> supposed provide a flexible resource management and task grouping
>>> ability, which somewhat implies that there cannot be two different
>>> resource managers per node without one knowing well the
>>> desires/goals/methods of the other and vice versa. And should there be
>>> only one manager per node - probably it will be easier for it to use
>>> cgroup subsystem directly rather than using a wrapper library?
>>>
>>
>> I disagree. Allowing multiple resource managers allows more flexibility.
>> One thing the configuration subsystem aims to do is to allow permissions
>> to the groups. With this happening, a resource manager does not need to
>> about the existence of other groups, it can control only the resources
>> allotted to it. And since the top level is controlled by the
>> administrator, he is aware of the groups and their needs. (I've given an
>> example of such a scenario in the document as well).
>>
> Imagine having a shared/joint household savings account with your wife, and
> taking money from it without your wife knowing and vice versa .. then at
> some point when you thought that you have some $5K to buy the new super
> duper laptop you dreamt about your entire life - surprise - no enough
> resources :)
> This is somewhat the equivalent of multiple independent resourse managers
> :) It won't end well :)
> Should they be expected to be adequate in doing their job, they cannot be
> independent since they manage a shared resource.
>

I don't quite agree with your analogy here. The point here is that each
resource manager operates in its own area, and has already been assigned
some resources which it cannot change. Its more like you can take $x at
the most from the account and your wife $y with x+y<=total money.

--
regards,
Dhaval

_____
Containers mailing list
Containers@lists.linux-foundation.org

Subject: Re: [RFC] libcg: design and plans
Posted by Paul Menage on Wed, 05 Mar 2008 18:55:02 GMT
View Forum Message <> Reply to Message

On Wed, Mar 5, 2008 at 6:24 AM, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
> Paul Menage wrote:
> > On Wed, Mar 5, 2008 at 3:07 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
> >> OK. Hmm, I've not really thought about it. At first thought, it should
> >> not be very difficult. Only thing I am not sure is the arbitrary
> >> grouping of the groups (ok, a bit confusing).
> >
> > I suspect that the main form of composite grouping is going to be
> > between parents and children. E.g. you might want to say things like:
> >
> > create_group(A, memory=1G, cpu=100)
> > create_group(B, parent=A, memory=inherit, cpu=20)
> > create_group(C, parent=A, memory=inherit, cpu=30)
> >
> > i.e. both B and C inherit/share their memory limit from their parent,
> > but have their own CPU groups (child groups of their parent?)
> >
>
> No, we don't plan on doing that. What we plan on doing is
>
> 1. Specify the mount point for each controller
> 2. In the create group API, specify the name of the group and the various
> parameters.
>
> If for example CPU is mounted at /cpu and Memory at /mem
>
> Then a specification for creation of group A would be of the form
>
> create_group(A, cpu=100, memory=100M)
>
> Then,
>
> /cpu/A has shares set to 100 and /mem/A has memory.limit set to 100M
>
> If you want to create subgroups under A, you specify
>
> create_group(A/B, memory=200M, cpu=50)
>
> That would create /cpu/A/B and /mem/A/B
>
> Please note that memory and CPU hierarchy needs work in the kernel. The shares

> and hierarchy support is pending. We need to make the res_counters
> infrastructure aware of hierarchies.
>

I think there are two different kinds of sharing going on here:

- A and B each have individual limits, and you additionally want their total usage to be capped by some parent limit. E.g. A and B each have a 100MB memory limit, and you want their total combined usage to not exceed 150MB. This kind of sharing has to be handled by the resource counter abstraction

- you want A and B to be treated identically for the purposes of some particular resource, e.g. you want a single CFS group to which all threads in A and B are equal members, or a single memory cgroup for all allocations by A or B, or the same device control table for A and B (but you want A and B to be treated separately for some other resource type). This can be handled in userspace in the way I outlined above, and it would be good if libcg could handle the setup required for this. It could also be done in the kernel with something like the parent/child subsystem group inheritance that I also mentioned above, if there was demand. But if so it should be a property of cgroups rather than any individual resource controller, since it's a feature that could be useful for all cgroups.

Paul

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by xpl on Wed, 05 Mar 2008 19:36:20 GMT
View Forum Message <> Reply to Message

Hi Dhaval,

Dhaval Giani ??????:
>> Imagine having a shared/joint household savings account with your wife, and
>> taking money from it without your wife knowing and vice versa .. then at
>> some point when you thought that you have some $5K to buy the new super
>> duper laptop you dreamt about your entire life - surprise - no enough
>> resources :)
>> This is somewhat the equivalent of multiple independent resourse managers
>> :) It won't end well :)
>> Should they be expected to be adequate in doing their job, they cannot be
>> independent since they manage a shared resource.

>>
>>
>
> I don't quite agree with your analogy here. The point here is that each
> resource manager operates in its own area, and has already been assigned
> some resources which it cannot change. Its more like you can take $x at
> the most from the account and your wife $y with x+y<=total money.
>
Ok .. so imagine that your kid got sick and you need more than $x, while
you wife does not need any money at that particular moment?
Would you:

a) take a loan (buy more hardware/resources, just because we can),
despite the fact you already have them in you account, and since we're
talking about your child it's in the interest of both you and your wife
(tha family, 'the whole' so to speak) to cure the illness

b) notify your wife there is an emergency and you will need some extra
money which has to come from her $y quota, take the money, make the kid
happy, and just continue business as usual? (that is - being smart and
dynamic in resource allocation)

With the proposed libcg, answer a) seems to be the only option .. and my
company is not like M$ so I don't have extra resources to waste just
because I was told resources cannot be managed dynamicaly :)
Why would one need to manage node resources dynamicaly: in our real-life
production system we have to manage resources dynamicaly, because any
other solution would require at least twice as much hardware, which will
also inevitably lead to a necessity to hire more qualified admins, which
is once again not so wise for small/medium business .. and not to
mention the extra CO2 caused by the few more dozens of servers :)

Regards,
Peter.

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

---

Subject: Re: [RFC] libcg: design and plans
Posted by Rik van Riel on Thu, 20 Mar 2008 22:04:55 GMT
View Forum Message <> Reply to Message

On Tue, 4 Mar 2008 20:53:41 +0530
Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:

> Hi,

>
> We have been working on a library for control groups which would provide
> simple APIs for programmers to utilize from userspace and make use of
> control groups.

Since somebody (*cough*openvz*cough*) will no doubt add security and
network separation to the control groups at some point in the future,
why not build on libvirt?

That way admins can also use the same tools for monitoring and managing
resource groups that they use for virtual machines.

http://libvirt.org/

--
All Rights Reversed

_____

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