Subject: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 02:16:08 GMT
View Forum Message <> Reply to Message

Containers:

Commodity HW is becoming more powerful. This is giving opportunity to run different workloads on the same platform for better HW resource utilization. To run different workloads efficiently on the same platform, it is critical that we have a notion of limits for each workload in Linux kernel. Current cpuset feature in Linux kernel provides grouping of CPU and memory support to some extent (for NUMA machines).

For example, a user can run a batch job like backup inside containers. This job if run unconstrained could step over most of the memory present in system thus impacting other workloads running on the system at that time. But when the same job is run inside containers then the backup job is run within container limits.

We use the term container to indicate a structure against which we track and charge utilization of system resources like memory, tasks etc for a workload. Containers will allow system admins to customize the underlying platform for different applications based on their performance and HW resource utilization needs. Containers contain enough infrastructure to allow optimal resource utilization without bogging down rest of the kernel. A system admin should be able to create, manage and free containers easily.

At the same time, changes in kernel are minimized so as this support can be easily integrated with mainline kernel.

The user interface for containers is through configfs. Appropriate file system privileges are required to do operations on each container. Currently implemented container resources are automatically visible to user space through /configfs/container/<container_name> after a container is created.

Signed-off-by: Rohit Seth <rohitseth@google.com>

Diffstat for the patch set (against linux-2.6.18-rc6-mm2_:

```
 Documentation/containers.txt |   65 ++++
 fs/inode.c                   |    3
 include/linux/container.h    |  167 ++++++++++
 include/linux/fs.h           |    5
 include/linux/mm_inline.h    |    4
 include/linux/mm_types.h     |    4
```

```
 include/linux/sched.h       |    6
 init/Kconfig                |    8
 kernel/Makefile             |    1
 kernel/container_configfs.c |  440 ++++++++++++++++++++++++++++++
 kernel/exit.c               |    2
 kernel/fork.c               |    9
 mm/Makefile                 |    2
 mm/container.c              |  658 ++++++++++++++++++++++++++++++++++++++++++++++
 mm/container_mm.c           |  512 ++++++++++++++++++++++++++++++++
 mm/filemap.c                |    4
 mm/page_alloc.c             |    3
 mm/rmap.c                   |    8
 mm/swap.c                   |    1
 mm/vmscan.c                 |    1
 20 files changed, 1902 insertions(+), 1 deletion(-)
```

Changes from version 1:
Fixed the Documentation error
Fixed the corruption in container task list
Added the support for showing all the tasks belonging to a container
through showtask attribute
moved the Kconfig changes to init directory (from mm)
Fixed the bug of unregistering container subsystem if we are not able to
create workqueue
Better support for handling limits for file pages. This now includes
support for flushing and invalidating page cache pages.
Minor other changes.

********** *****
This patch set has basic container support that includes:

- Create a container using mkdir command in configfs

- Free a container using rmdir command

- Dynamically adjust memory and task limits for container.

- Add/Remove a task to container (given a pid)

- Files are currently added as part of open from a task that already
belongs to a container.

- Keep track of active, anonymous, mapped and pagecache usage of
container memory

- Does not allow more than task_limit number of tasks to be created in
the container.

- Over the limit memory handler is called when number of pages (anon + pagecache) exceed the limit. It is also called when number of active pages exceed the page limit. Currently, this memory handler scans the mappings and tasks belonging to container (file and anonymous) and tries to deactivate pages. If the number of page cache pages is also high then it also invalidate mappings. The thought behind this scheme is, it is okay for containers to go over limit as long they run in degraded manner when they are over their limit. Also, if there is any memory pressure then pages belonging to over the limit container(s) become prime candidates for kernel reclaimer. Container mutex is also held during the time this handler is working its way through to prevent any further addition of resources (like tasks or mappings) to this container. Though it also blocks removal of same resources from the container for the same time. It is possible that over the limit page handler takes lot of time if memory pressure on a container is continuously very high. The limits, like how long a task should schedule out when it hits memory limit, is also on the lower side at present (particularly when it is memory hogger). But should be easy to change if need be.

- Indicate the number of times the page limit and task limit is hit

- Indicate the tasks (pids) belonging to container.

Below is a one line description for patches that will follow:

[patch01]: Documentation on how to use containers (Documentation/container.txt)

[patch02]: Changes in the generic part of kernel code

[patch03]: Container's interface with configfs

[patch04]: Core container support

[patch05]: Over the limit memory handler.

TODO:

- some code(like container_add_task) in mm/container.c should go elsewhere.
- Support adding/removing a file name to container through configfs
- /proc/pid/container to show the container id (or name)
- More testing for memory controller. Currently it is possible that limits are exceeded. See if a call to reclaim can be easily integrated.
- Kernel memory tracking (based on patches from BC)
- Limit on user locked memory
- Huge memory support

- Stress testing with containers
- One shot view of all containers
- CKRM folks are interested in seeing all processes belonging to a container. Add the attribute show_tasks to container.
- Add logic so that the sum of limits are not exceeding appropriate system requirements.
- Extend it with other controllers (CPU and Disk I/O)
- Add flags bits for supporting different actions (like in some cases provide a hard memory limit and in some cases it could be soft).
- Capability to kill processes for the extreme cases.
...

This is based on lot of discussions over last month or so. I hope this patch set is something that we can agree and more support can be added on top of this. Please provide feedback and add other extensions that are useful in the TODO list.

Thanks,
-rohit

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Nick Piggin on Wed, 20 Sep 2006 05:39:00 GMT
View Forum Message <> Reply to Message

Rohit Seth wrote:

>Containers:
>
>
[...]

>This is based on lot of discussions over last month or so. I hope this
>patch set is something that we can agree and more support can be added
>on top of this. Please provide feedback and add other extensions that
>are useful in the TODO list.
>

Hi Rohit,

Sorry for the late reply. I was just about to comment on your earlier patchset but I will do so here instead.

Anyway I don't think I have much to say other than: this is almost exactly as I had imagined the memory resource tracking should look like. Just a small number of hooks and a very simple set of rules for tracking allocations. Also, the possibility to track kernel allocations as a whole rather than at individual callsites (which

shouldn't be too difficult to implement).

If anything I would perhaps even argue for further cutting down the
number of hooks and add them back as they prove to be needed.

I'm not sure about containers & workload management people, but from
a core mm/ perspective I see no reason why this couldn't get in,
given review and testing. Great!

Nick
--

Send instant messages to your online friends http://au.messenger.yahoo.com

---

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Cedric Le Goater on Wed, 20 Sep 2006 13:06:59 GMT
View Forum Message <> Reply to Message

Rohit Seth wrote:
>
> This is based on lot of discussions over last month or so. I hope this
> patch set is something that we can agree and more support can be added
> on top of this. Please provide feedback and add other extensions that
> are useful in the TODO list.

thanks rohit for this patchset.

I applied it on rc7-mm1, compiles smoothly and boots. Here's a edge
issue, I got this oops while rmdir a container with running tasks.

C.

BUG: unable to handle kernel paging request at virtual address 00100100
printing eip:
c01470ea
*pde = 00000000
Oops: 0000 [#1]
last sysfs file: /block/hda/range
Modules linked in:
CPU: 0
EIP: 0060:[<c01470ea>] Not tainted VLI
EFLAGS: 00000282 (2.6.18-rc7-mm1-lxc2 #8)
EIP is at container_remove_tasks+0x21/0x3c
eax: 00000007 ebx: 000ffbb0 ecx: c13e4330 edx: 00000196
esi: c13e42f4 edi: 00000000 ebp: c6e39eb8 esp: c6e39eb0
ds: 007b es: 007b ss: 0068

Process rmdir (pid: 9081, ti=c6e38000 task=c7223710 task.ti=c6e38000)
Stack: c13e42f4 00000000 c6e39ec8 c0147206 c13e42c0 c0307e44 c6e39ed4 c012e060
c13e42c0 c6e39eec c017c0ac 00000000 c13e42d8 c017c0cb c13e42c0 c6e39ef4
c017c0d6 c6e39f04 c01f441e c0307de0 c0307de0 c6e39f0c c017c068 c6e39f30
Call Trace:
[<c0147206>] free_container+0x18/0x95
[<c012e060>] simple_containerfs_release+0x15/0x21
[<c017c0ac>] config_item_cleanup+0x42/0x61
[<c017c0d6>] config_item_release+0xb/0xd
[<c01f441e>] kref_put+0x66/0x74
[<c017c068>] config_item_put+0x14/0x16
[<c017b760>] configfs_rmdir+0x16d/0x1b5
[<c014ff07>] vfs_rmdir+0x54/0x96
[<c015180c>] do_rmdir+0x8f/0xcc
[<c0151888>] sys_rmdir+0x10/0x12
[<c01027dc>] syscall_call+0x7/0xb

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 16:25:03 GMT
View Forum Message <> Reply to Message

On Tue, 19 Sep 2006, Rohit Seth wrote:

> For example, a user can run a batch job like backup inside containers.
> This job if run unconstrained could step over most of the memory present
> in system thus impacting other workloads running on the system at that
> time. But when the same job is run inside containers then the backup
> job is run within container limits.

I just saw this for the first time since linux-mm was not cced. We have
discussed a similar mechanism on linux-mm.

We already have such a functionality in the kernel its called a cpuset. A
container could be created simply by creating a fake node that then
allows constraining applications to this node. We already track the
types of pages per node. The statistics you want are already existing.
See /proc/zoneinfo and /sys/devices/system/node/node*/*.

> We use the term container to indicate a structure against which we track
> and charge utilization of system resources like memory, tasks etc for a
> workload. Containers will allow system admins to customize the
> underlying platform for different applications based on their
> performance and HW resource utilization needs. Containers contain
> enough infrastructure to allow optimal resource utilization without
> bogging down rest of the kernel. A system admin should be able to
> create, manage and free containers easily.

Right thats what cpusets do and it has been working fine for years. Maybe
Paul can help you if you find anything missing in the existing means to
control resources.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 16:26:41 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Nick Piggin wrote:

> I'm not sure about containers & workload management people, but from
> a core mm/ perspective I see no reason why this couldn't get in,
> given review and testing. Great!

Nack. We already have the ability to manage workloads. We may want to
extend the existing functionality but this is duplicating what is already
available through cpusets.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 16:27:46 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 15:39 +1000, Nick Piggin wrote:

> Anyway I don't think I have much to say other than: this is almost
> exactly as I had imagined the memory resource tracking should look
> like. Just a small number of hooks and a very simple set of rules for
> tracking allocations. Also, the possibility to track kernel
> allocations as a whole rather than at individual callsites (which
> shouldn't be too difficult to implement).
>

I've started looking in that direction. First shot could just be
tracking kernel memory consumption w/o worrying about whether it is slab
or PT etc. Hopefully next patchset will have that support integrated.

> If anything I would perhaps even argue for further cutting down the
> number of hooks and add them back as they prove to be needed.
>

I think the current set of changes (and tracking of different
components) is necessary for memory handler to do the right thing. Plus
it is possible that user land management tools can also make use of this
information.

> I'm not sure about containers & workload management people, but from
> a core mm/ perspective I see no reason why this couldn't get in,
> given review and testing. Great!
>

That is great to know. Thanks. Hopefully it is getting enough coverage
to get there.

-rohit

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Nick Piggin on Wed, 20 Sep 2006 16:44:04 GMT
View Forum Message <> Reply to Message

On Wed, Sep 20, 2006 at 09:25:03AM -0700, Christoph Lameter wrote:
> On Tue, 19 Sep 2006, Rohit Seth wrote:
>
> > > For example, a user can run a batch job like backup inside containers.
> > > This job if run unconstrained could step over most of the memory present
> > > in system thus impacting other workloads running on the system at that
> > > time. But when the same job is run inside containers then the backup
> > > job is run within container limits.
>
> I just saw this for the first time since linux-mm was not cced. We have
> discussed a similar mechanism on linux-mm.
>
> We already have such a functionality in the kernel its called a cpuset. A
> container could be created simply by creating a fake node that then
> allows constraining applications to this node. We already track the
> types of pages per node. The statistics you want are already existing.
> See /proc/zoneinfo and /sys/devices/system/node/node*/*.
>
> > > We use the term container to indicate a structure against which we track
> > > and charge utilization of system resources like memory, tasks etc for a
> > > workload. Containers will allow system admins to customize the
> > > underlying platform for different applications based on their
> > > performance and HW resource utilization needs. Containers contain
> > > enough infrastructure to allow optimal resource utilization without
> > > bogging down rest of the kernel. A system admin should be able to
> > > create, manage and free containers easily.
>
> Right thats what cpusets do and it has been working fine for years. Maybe
> Paul can help you if you find anything missing in the existing means to
> control resources.

What I like about Rohit's patches is the page tracking stuff which

seems quite simple but capable.

I suspect cpusets don't quite provide enough features for non-exclusive
use of memory (eg. page tracking for directed reclaim).

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 16:45:43 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 15:06 +0200, Cedric Le Goater wrote:
> Rohit Seth wrote:
> >
> > This is based on lot of discussions over last month or so. I hope this
> > patch set is something that we can agree and more support can be added
> > on top of this. Please provide feedback and add other extensions that
> > are useful in the TODO list.
>
> thanks rohit for this patchset.
>
> I applied it on rc7-mm1, compiles smoothly and boots. Here's a edge
> issue, I got this oops while rmdir a container with running tasks.
>

Thanks for pointing this out. I will look into this and send the update.

-rohit

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 16:48:13 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Nick Piggin wrote:

> > Right thats what cpusets do and it has been working fine for years. Maybe
> > Paul can help you if you find anything missing in the existing means to
> > control resources.
>
> What I like about Rohit's patches is the page tracking stuff which
> seems quite simple but capable.
>
> I suspect cpusets don't quite provide enough features for non-exclusive
> use of memory (eg. page tracking for directed reclaim).

Look at the VM statistics please. We have detailed page statistics per
zone these days. If there is anything missing then this would best be put

into general functionality. When I looked at it, I saw page statistics
that were replicating things that we already track per zone. All these
would become available if a container is realized via a node and we would
be using proven VM code.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Nick Piggin on Wed, 20 Sep 2006 16:56:11 GMT
View Forum Message <> Reply to Message

Christoph Lameter wrote:
> On Wed, 20 Sep 2006, Nick Piggin wrote:
>
>
>>I'm not sure about containers & workload management people, but from
>>a core mm/ perspective I see no reason why this couldn't get in,
>>given review and testing. Great!
>
>
> Nack. We already have the ability to manage workloads. We may want to
> extend the existing functionality but this is duplicating what is already
> available through cpusets.

If it wasn't clear was talking specifically about the hooks for page
tracking rather than the whole patchset. If anybody wants such page
tracking infrastructure in the kernel, then this (as opposed to the
huge beancounters stuff) is what it should look like.

But as I said above, I don't know what the containers and workload
management people want exactly... The recent discussions about using
nodes and cpusets for memory workload management does seem like a
promising idea, and if it would avoid the need for this kind of
per-page tracking entirely, then that would probably be even better.

--
SUSE Labs, Novell Inc.
Send instant messages to your online friends http://au.messenger.yahoo.com

---

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Nick Piggin on Wed, 20 Sep 2006 17:00:37 GMT
View Forum Message <> Reply to Message

(this time to the lists as well)

Peter Zijlstra wrote:

> I'd much rather containterize the whole reclaim code, which should not
> be too hard since he already adds a container pointer to struct page.

Yes, and I tend to agree with you. I probably wasn't clear, but I was mainly talking about just the memory resource tracking part of this patchset.

I am less willing to make a judgement about reclaim, because I don't know very much about the workloads or the guarantees they attempt to provide.

> Esp. when we get some of my page reclaim abstractions merged, moving the
> reclaim from struct zone to a container is not a lot of work. (this is
> basically what one of the ckrm mm policies did too)

I do agree that it would be nicer to not have a completely different scheme for doing their own page reclaim, but rather use the existing code (*provided* that it is designed in the same, minimally intrusive manner as the page tracking).

I can understand how it is attractive to create a new subsystem to solve a particular problem, but once it is in the kernel it has to be maintained regardless, so if it can be done in a way that shares more of the current infrastructure (nicely) then that would be a better solution.

I like that they're investigating the use of memory nodes for this. It seems like the logical starting place.

> I still have to reread what Rohit does for file backed pages, that gave
> my head a spin.
> I've been thinking a bit on that problem, and it would be possible to
> share all address_space pages equally between attached containers, this
> would lose some accuracy, since one container could read 10% of the file
> and another 90%, but I don't think that is a common scenario.

Yeah, I'm not sure about that. I don't think really complex schemes are needed... but again I might need more knowledge of their workloads and problems.

--
SUSE Labs, Novell Inc.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Nick Piggin on Wed, 20 Sep 2006 17:07:35 GMT
View Forum Message <> Reply to Message

On Wed, Sep 20, 2006 at 09:48:13AM -0700, Christoph Lameter wrote:
> On Wed, 20 Sep 2006, Nick Piggin wrote:
>
> > > Right thats what cpusets do and it has been working fine for years. Maybe
> > > Paul can help you if you find anything missing in the existing means to
> > > control resources.
> >
> > What I like about Rohit's patches is the page tracking stuff which
> > seems quite simple but capable.
> >
> > I suspect cpusets don't quite provide enough features for non-exclusive
> > use of memory (eg. page tracking for directed reclaim).
>
> Look at the VM statistics please. We have detailed page statistics per
> zone these days. If there is anything missing then this would best be put
> into general functionality. When I looked at it, I saw page statistics
> that were replicating things that we already track per zone. All these
> would become available if a container is realized via a node and we would
> be using proven VM code.

Look at what the patches do. These are not only for hard partitioning
of memory per container but also those that share memory (eg. you might
want each to share 100MB of memory, up to a max of 80MB for an individual
container).

The nodes+cpusets stuff doesn't seem to help with that because you
with that because you fundamentally need to track pages on a per
container basis otherwise you don't know who's got what.

Now if, in practice, it turns out that nobody really needed these
features then of course I would prefer the cpuset+nodes approach. My
point is that I am not in a position to know who wants what, so I
hope people will come out and discuss some of these issues.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 17:08:00 GMT
View Forum Message <> Reply to Message

On Thu, 21 Sep 2006, Nick Piggin wrote:

> If it wasn't clear was talking specifically about the hooks for page
> tracking rather than the whole patchset. If anybody wants such page
> tracking infrastructure in the kernel, then this (as opposed to the

> huge beancounters stuff) is what it should look like.

Could you point to the patch and a description for what is meant here by
page tracking (did not see that in the patch, maybe I could not find it)?
If these are just statistics then we likely already have
them.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Alan Cox on Wed, 20 Sep 2006 17:10:57 GMT
View Forum Message <> Reply to Message

Ar Mer, 2006-09-20 am 09:25 -0700, ysgrifennodd Christoph Lameter:
> We already have such a functionality in the kernel its called a cpuset. A
> container could be created simply by creating a fake node that then
> allows constraining applications to this node. We already track the
> types of pages per node. The statistics you want are already existing.
> See /proc/zoneinfo and /sys/devices/system/node/node*/*.

CPUsets don't appear to scale to large numbers of containers (say 5000,
with 200-500 doing stuff at a time). They also don't appear to do any
tracking of kernel side resource objects, which is critical to
containment. Indeed for some of us the CPU management and user memory
management angle is mostly uninteresting.

I'm also not clear how you handle shared pages correctly under the fake
node system, can you perhaps explain that further how this works for say
a single apache/php/glibc shared page set across 5000 containers each a
web site.

Alan

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 17:12:27 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Nick Piggin wrote:

> Look at what the patches do. These are not only for hard partitioning
> of memory per container but also those that share memory (eg. you might
> want each to share 100MB of memory, up to a max of 80MB for an individual
> container).

So far I have not been able to find the hooks to the VM. The sharing
would also work with nodes. Just create a couple of nodes with the sizes you
want and then put the node with the shared memory into the cpusets for the

apps sharing them.

> The nodes+cpusets stuff doesn't seem to help with that because you
> with that because you fundamentally need to track pages on a per
> container basis otherwise you don't know who's got what.

Hmmm... That gets into issues of knowing how many pages are in use by an application and that is fundamentally difficult to do due to pages being shared between processes.

> Now if, in practice, it turns out that nobody really needed these
> features then of course I would prefer the cpuset+nodes approach. My
> point is that I am not in a position to know who wants what, so I
> hope people will come out and discuss some of these issues.

I'd really be interested to have proper tracking of memory use for processes but I am not sure what use the containers would be there.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Alan Cox on Wed, 20 Sep 2006 17:12:45 GMT
View Forum Message <> Reply to Message

Ar Iau, 2006-09-21 am 03:00 +1000, ysgrifennodd Nick Piggin:
> > I've been thinking a bit on that problem, and it would be possible to
> > share all address_space pages equally between attached containers, this
> > would lose some accuracy, since one container could read 10% of the file
> > and another 90%, but I don't think that is a common scenario.
>
>
> Yeah, I'm not sure about that. I don't think really complex schemes
> are needed... but again I might need more knowledge of their workloads
> and problems.

Any scenario which permits "cheating" will be a scenario that happens because people will try and cheat.

Alan

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 17:15:52 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Alan Cox wrote:

> Ar Mer, 2006-09-20 am 09:25 -0700, ysgrifennodd Christoph Lameter:

> > We already have such a functionality in the kernel its called a cpuset. A
> > container could be created simply by creating a fake node that then
> > allows constraining applications to this node. We already track the
> > types of pages per node. The statistics you want are already existing.
> > See /proc/zoneinfo and /sys/devices/system/node/node*/*.
>
> CPUsets don't appear to scale to large numbers of containers (say 5000,
> with 200-500 doing stuff at a time). They also don't appear to do any
> tracking of kernel side resource objects, which is critical to
> containment. Indeed for some of us the CPU management and user memory
> management angle is mostly uninteresting.

The scalability issues can certainly be managed. See the discussions on
linux-mm. Kernel side resource objects? slab pages? Those are tracked.

> I'm also not clear how you handle shared pages correctly under the fake
> node system, can you perhaps explain that further how this works for say
> a single apache/php/glibc shared page set across 5000 containers each a
> web site.

Cpusets can share nodes. I am not sure what the problem would be? Paul may
be able to give you more details.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Alan Cox on Wed, 20 Sep 2006 17:16:23 GMT
View Forum Message <> Reply to Message

Ar Iau, 2006-09-21 am 02:56 +1000, ysgrifennodd Nick Piggin:
> But as I said above, I don't know what the containers and workload
> management people want exactly... The recent discussions about using
> nodes and cpusets for memory workload management does seem like a
> promising idea, and if it would avoid the need for this kind of
> per-page tracking entirely, then that would probably be even better.

I think you can roughly break it down to

- I want one group of users not to be able to screw another group of
users or the box but don't care about anything else. The basic
beancounter stuff handles this. Generally they also want maximal
sharing.

- I want to charge people for portions of machine use (mostly accounting
and some fairness)

- I don't want any user to be able to hog the system to the harm of the
others but don't care about overcommit of idle resources (think about
the 5000 apaches on a box case)

- I want to be able to divide resources reasonably accurately all the
time between groups of users

Alan

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Nick Piggin on Wed, 20 Sep 2006 17:19:44 GMT
View Forum Message <> Reply to Message

Christoph Lameter wrote:
> On Thu, 21 Sep 2006, Nick Piggin wrote:
>
>
>>If it wasn't clear was talking specifically about the hooks for page
>>tracking rather than the whole patchset. If anybody wants such page
>>tracking infrastructure in the kernel, then this (as opposed to the
>>huge beancounters stuff) is what it should look like.
>
>
> Could you point to the patch and a description for what is meant here by
> page tracking (did not see that in the patch, maybe I could not find it)?
> If these are just statistics then we likely already have
> them.
>

Patch 2/5 in this series provides hooks, and they are pretty unintrusive.

```
 mm/filemap.c      |    4 ++++
 mm/page_alloc.c   |    3 +++
 mm/rmap.c         |    8 +++++++-
 mm/swap.c         |    1 +
 mm/vmscan.c       |    1 +
```

--
SUSE Labs, Novell Inc.
Send instant messages to your online friends http://au.messenger.yahoo.com

---

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Paul Menage on Wed, 20 Sep 2006 17:23:54 GMT
View Forum Message <> Reply to Message

On 9/20/06, Nick Piggin <nickpiggin@yahoo.com.au> wrote:

> Yeah, I'm not sure about that. I don't think really complex schemes

> are needed... but again I might need more knowledge of their workloads
> and problems.
>

The basic need for separating files into containers distinct from the
tasks that are using them arises when you have several "jobs" all
working with the same large data set. (Possibly read-only data files,
or possibly one job is updating a dataset that's being used by other
jobs).
For automated job-tracking and scheduling, it's important to be able
to distinguish shared usage from individual usage (e.g. to be able to
answer questions "if I kill job X, how much memory do I get back?" and
"how do I recover 1G of memory on this machine")

As an example, assume two jobs each with 100M of anonymous memory both
mapping the same 1G file, for a total usage of 1.2G.

Any setup that doesn't let you distinguish shared and private usage
makes it hard to answer that kind of scheduling questions. E.g.:

- first user gets charged for the page -> first job reported as 1.1G,
and the second as 0.1G.

- page charges get shared between all users of the page -> two tasks
using 0.6G each.

- all users get charged for the page -> two tasks using 1.1G each.

But in fact killing either one of these jobs individually would only
free up 100M

By explicitly letting userspace see that there are two jobs each with
a private usage of 100M, and they're sharing a dataset of 1G, it's
possible to make more informed decisions.

The issue of telling the kernel exactly which files/directories need
to be accounted separately can be handled by userspace.

It could be done by per-page accounting, or by constraining particular
files to particular memory zones, or by just tracking/limiting the
number of pages from each address_space in the pagecache, but I think
that it's important that the kernel at least provide the primitive
support for this.

Paul

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Alan Cox on Wed, 20 Sep 2006 17:24:07 GMT
View Forum Message <> Reply to Message

Ar Mer, 2006-09-20 am 10:15 -0700, ysgrifennodd Christoph Lameter:
> The scalability issues can certainly be managed. See the discussions on
> linux-mm.

I'll take a look at a web archive of it, I don't follow -mm.

> Kernel side resource objects? slab pages? Those are tracked.

Slab pages isn't a useful tracking tool for two reasons. The first is
that some resources are genuinely a shared kernel managed pool and
should be treated that way - thats obviously easy to sort out.

The second is that slab pages are not the granularity of allocations so
it becomes possible (and deliberately influencable) to make someone else
allocate the pages all the time so you don't pay the cost. Hence the
beancounters track the real objects.

> Cpusets can share nodes. I am not sure what the problem would be? Paul may
> be able to give you more details.

If it can do it in a human understandable way, configured at runtime
with dynamic sharing, overcommit and reconfiguration of sizes then
great. Lets see what Paul has to say.

Alan

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 17:26:47 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 09:25 -0700, Christoph Lameter wrote:
> On Tue, 19 Sep 2006, Rohit Seth wrote:
>
> > For example, a user can run a batch job like backup inside containers.
> > This job if run unconstrained could step over most of the memory present
> > in system thus impacting other workloads running on the system at that
> > time. But when the same job is run inside containers then the backup
> > job is run within container limits.
>
> I just saw this for the first time since linux-mm was not cced. We have
> discussed a similar mechanism on linux-mm.
>
> We already have such a functionality in the kernel its called a cpuset. A

> container could be created simply by creating a fake node that then
> allows constraining applications to this node. We already track the
> types of pages per node. The statistics you want are already existing.
> See /proc/zoneinfo and /sys/devices/system/node/node*/*.
>
> > We use the term container to indicate a structure against which we track
> > and charge utilization of system resources like memory, tasks etc for a
> > workload. Containers will allow system admins to customize the
> > underlying platform for different applications based on their
> > performance and HW resource utilization needs. Containers contain
> > enough infrastructure to allow optimal resource utilization without
> > bogging down rest of the kernel. A system admin should be able to
> > create, manage and free containers easily.
>
> Right thats what cpusets do and it has been working fine for years. Maybe
> Paul can help you if you find anything missing in the existing means to
> control resources.

cpusets provides cpu and memory NODES binding to tasks. And I think it
works great for NUMA machines where you have different nodes with its
own set of CPUs and memory. The number of those nodes on a commodity HW
is still 1. And they can have 8-16 CPUs and in access of 100G of
memory. You may start using fake nodes (untested territory) to
translate a single node machine into N different nodes. But am not sure
if this number of nodes can change dynamically on the running machine or
a reboot is required to change the number of nodes.

Though when you want to have in access of 100 containers then the cpuset
function starts popping up on the oprofile chart very aggressively. And
this is the cost that shouldn't have to be paid (particularly) for a
single node machine.

And what happens when you want to have cpuset with memory that needs to
be even further fine grained than each node.

Containers also provide a mechanism to move files to containers. Any
further references to this file come from the same container rather than
the container which is bringing in a new page.

In future there will be more handlers like CPU and disk that can be
easily embeded into this container infrastructure.

-rohit

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Paul Menage on Wed, 20 Sep 2006 17:30:24 GMT

On 9/20/06, Alan Cox <alan@lxorguk.ukuu.org.uk> wrote:
>
> I'm also not clear how you handle shared pages correctly under the fake
> node system, can you perhaps explain that further how this works for say
> a single apache/php/glibc shared page set across 5000 containers each a
> web site.

If you can associate files with containers, you can have a "shared libraries" container that the libraries/binaries for apache/php/glibc are associated with - all pages from those files are then accounted to the shared container. So you can see that there are 5000 apaches each using say 10MB privately, and sharing a container with 100MB of file data. This can also be O(1) in the number of apache containers.

Paul

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Nick Piggin on Wed, 20 Sep 2006 17:30:30 GMT

Alan Cox wrote:
> Ar Iau, 2006-09-21 am 03:00 +1000, ysgrifennodd Nick Piggin:
>
>> > I've been thinking a bit on that problem, and it would be possible to
>> > share all address_space pages equally between attached containers, this
>> > would lose some accuracy, since one container could read 10% of the file
>> > and another 90%, but I don't think that is a common scenario.
>>
>>
>>Yeah, I'm not sure about that. I don't think really complex schemes
>>are needed... but again I might need more knowledge of their workloads
>>and problems.
>
>
> Any scenario which permits "cheating" will be a scenario that happens
> because people will try and cheat.

That's true, and that's one reason why I've advocated the solution implemented by Rohit's patches, that is: just throw in the towel and be happy to count just pages.

Look at the beancounter stuff, and it has hooks (in the form of gfp flags) throughput the tree, and they still manage to miss accounting user exploitable memory overallocation from some callers. Maintaining that will be much more difficult and error prone.

--
SUSE Labs, Novell Inc.
Send instant messages to your online friends http://au.messenger.yahoo.com

---

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 17:30:36 GMT
View Forum Message <> Reply to Message

On Thu, 21 Sep 2006, Nick Piggin wrote:

> Patch 2/5 in this series provides hooks, and they are pretty unintrusive.

Ok. We shadow existing vm counters add stuff to the adress_space
structure. The task add / remove is duplicating what some of the cpuset
hooks do. That clearly shows that we are just duplicating functionality.

The mapping things are new.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 17:35:12 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Alan Cox wrote:

> If it can do it in a human understandable way, configured at runtime
> with dynamic sharing, overcommit and reconfiguration of sizes then
> great. Lets see what Paul has to say.

You have full VM support this means overcommit and every else you are used
ot.

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Paul Menage on Wed, 20 Sep 2006 17:37:19 GMT
View Forum Message <> Reply to Message

On 9/20/06, Rohit Seth <rohitseth@google.com> wrote:
>
> cpusets provides cpu and memory NODES binding to tasks. And I think it
> works great for NUMA machines where you have different nodes with its
> own set of CPUs and memory. The number of those nodes on a commodity HW
> is still 1. And they can have 8-16 CPUs and in access of 100G of
> memory. You may start using fake nodes (untested territory) to

I've been experimenting with this, and it's looking promising.

>
> Containers also provide a mechanism to move files to containers. Any
> further references to this file come from the same container rather than
> the container which is bringing in a new page.

This could be done with memory nodes too - a vma can specify its
memory policy, so binding individual files to nodes shouldn't be hard.

>
> In future there will be more handlers like CPU and disk that can be
> easily embeded into this container infrastructure.

I think that at least the userspace API for adding more handlers would
need to be defined before actually committing a container patch, even
if the kernel code isn't yet extensible. The CKRM/RG interface is a
good start towards this.

Paul

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 17:38:13 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Rohit Seth wrote:

> cpusets provides cpu and memory NODES binding to tasks. And I think it
> works great for NUMA machines where you have different nodes with its
> own set of CPUs and memory. The number of those nodes on a commodity HW
> is still 1. And they can have 8-16 CPUs and in access of 100G of
> memory. You may start using fake nodes (untested territory) to

See linux-mm. We just went through a series of tests and functionality
wise it worked just fine.

> translate a single node machine into N different nodes. But am not sure
> if this number of nodes can change dynamically on the running machine or
> a reboot is required to change the number of nodes.

This is commonly discussed under the subject of memory hotplug.

> Though when you want to have in access of 100 containers then the cpuset
> function starts popping up on the oprofile chart very aggressively. And
> this is the cost that shouldn't have to be paid (particularly) for a
> single node machine.

Yes this is a new way of using cpusets but it works and we could fix the scalability issues rather than adding new subsystems.

> And what happens when you want to have cpuset with memory that needs to
> be even further fine grained than each node.

New node?

> Containers also provide a mechanism to move files to containers. Any
> further references to this file come from the same container rather than
> the container which is bringing in a new page.

Hmmmm... Thats is interesting.

> In future there will be more handlers like CPU and disk that can be
> easily embeded into this container infrastructure.

I think we should have one container mechanism instead of multiple. Maybe merge the two? The cpuset functionality is well established and working right.

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Paul Menage on Wed, 20 Sep 2006 17:42:24 GMT
View Forum Message <> Reply to Message

On 9/20/06, Christoph Lameter <clameter@sgi.com> wrote:
>
> I think we should have one container mechanism instead of multiple. Maybe
> merge the two? The cpuset functionality is well established and working
> right.

The basic container abstraction provided by cpusets is very nice - maybe rename it from "cpuset" to "container"? Since it already provides access to memory nodes as well as cpus, and could be extended to handle other resource types too (network, disk).

Paul

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 17:50:57 GMT
View Forum Message <> Reply to Message

On Thu, 2006-09-21 at 03:00 +1000, Nick Piggin wrote:
> (this time to the lists as well)
>

> Peter Zijlstra wrote:
>
> > I'd much rather containterize the whole reclaim code, which should not
> > be too hard since he already adds a container pointer to struct page.
>
>

Right now the memory handler in this container subsystem is written in
such a way that when existing kernel reclaimer kicks in, it will first
operate on those (container with pages over the limit) pages first. But
in general I like the notion of containerizing the whole reclaim code.

> > I still have to reread what Rohit does for file backed pages, that gave
> > my head a spin.

Please let me know if there is any specific part that isn't making much
sense.

-rohit

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 17:52:12 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Rohit Seth wrote:

> Right now the memory handler in this container subsystem is written in
> such a way that when existing kernel reclaimer kicks in, it will first
> operate on those (container with pages over the limit) pages first. But
> in general I like the notion of containerizing the whole reclaim code.

Which comes naturally with cpusets.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Nick Piggin on Wed, 20 Sep 2006 18:03:28 GMT
View Forum Message <> Reply to Message

Christoph Lameter wrote:
> On Thu, 21 Sep 2006, Nick Piggin wrote:
>
>
>>Patch 2/5 in this series provides hooks, and they are pretty unintrusive.
>
>
> Ok. We shadow existing vm counters add stuff to the adress_space

> structure. The task add / remove is duplicating what some of the cpuset
> hooks do. That clearly shows that we are just duplicating functionality.

I don't think so. To start with, the point about containers is they are
not per address_space.

But secondly, these are hooks from the container subsystem into the mm
subsystem. As such, they might do something a bit more or different
than simple statistics, and we don't want to teach the core mm/ about
what that might be. You also want to be able to configure them out
entirely.

I think it is fine to add some new hooks in fundamental (ie mm agnostic)
points. Without getting to the fine details about exactly how the hooks
are implemented, or what information needs to be tracked, I think we can
say that they are not much burden for mm/ to bear (if they turn out to
be usable).

--
SUSE Labs, Novell Inc.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Peter Zijlstra on Wed, 20 Sep 2006 18:06:26 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 10:52 -0700, Christoph Lameter wrote:
> On Wed, 20 Sep 2006, Rohit Seth wrote:
>
> > > Right now the memory handler in this container subsystem is written in
> > > such a way that when existing kernel reclaimer kicks in, it will first
> > > operate on those (container with pages over the limit) pages first. But
> > > in general I like the notion of containerizing the whole reclaim code.
>
> Which comes naturally with cpusets.

How are shared mappings dealt with, are pages charged to the set that
first faults them in?

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 18:07:58 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 10:38 -0700, Christoph Lameter wrote:
> On Wed, 20 Sep 2006, Rohit Seth wrote:

>
> > cpusets provides cpu and memory NODES binding to tasks. And I think it
> > works great for NUMA machines where you have different nodes with its
> > own set of CPUs and memory. The number of those nodes on a commodity HW
> > is still 1. And they can have 8-16 CPUs and in access of 100G of
> > memory. You may start using fake nodes (untested territory) to
>
> See linux-mm. We just went through a series of tests and functionality
> wise it worked just fine.
>

I thought the fake NUMA support still does not work on x86_64 baseline
kernel. Though Paul and Andrew have patches to make it work.

> > translate a single node machine into N different nodes. But am not sure
> > if this number of nodes can change dynamically on the running machine or
> > a reboot is required to change the number of nodes.
>
> This is commonly discussed under the subject of memory hotplug.
>

So now we depend on getting memory hot-plug to work for faking up these
nodes ...for the memory that is already present in the system. It just
does not sound logical.

> > Though when you want to have in access of 100 containers then the cpuset
> > function starts popping up on the oprofile chart very aggressively. And
> > this is the cost that shouldn't have to be paid (particularly) for a
> > single node machine.
>
> Yes this is a new way of using cpusets but it works and we could fix the
> scalability issues rather than adding new subsystems.
>

I think when you have 100's of zones then cost of allocating a page will
include checking cpuset validation and different zone list traversals
and checks...unless there is major surgery.

> > And what happens when you want to have cpuset with memory that needs to
> > be even further fine grained than each node.
>
> New node?
>

Am not clear how is this possible. Could you or Paul please explain.

> > Containers also provide a mechanism to move files to containers. Any
> > further references to this file come from the same container rather than

> > the container which is bringing in a new page.
>
> Hmmmm... Thats is interesting.
>
> > In future there will be more handlers like CPU and disk that can be
> > easily embeded into this container infrastructure.
>
> I think we should have one container mechanism instead of multiple. Maybe
> merge the two? The cpuset functionality is well established and working
> right.
>

I agree that we will need one container subsystem in the long run.
Something that can easily adapt to different configurations.

-rohit

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 18:14:59 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 20:06 +0200, Peter Zijlstra wrote:
> On Wed, 2006-09-20 at 10:52 -0700, Christoph Lameter wrote:
> > On Wed, 20 Sep 2006, Rohit Seth wrote:
> >
> > > Right now the memory handler in this container subsystem is written in
> > > such a way that when existing kernel reclaimer kicks in, it will first
> > > operate on those (container with pages over the limit) pages first. But
> > > in general I like the notion of containerizing the whole reclaim code.
> >
> > Which comes naturally with cpusets.
>
> How are shared mappings dealt with, are pages charged to the set that
> first faults them in?
>

For anonymous pages (simpler case), they get charged to the faulting
task's container.

For filesystem pages (could be shared across tasks running different
containers): Every time a new file mapping is created, it is bound to a
container of the process creating that mapping. All subsequent pages
belonging to this mapping will belong to this container, irrespective of
different tasks running in different containers accessing these pages.
Currently, I've not implemented a mechanism to allow a file to be
specifically moved into or out of container. But when that gets
implemented then all pages belonging to a mapping will also move out of

container (or into a new container).

-rohit

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Peter Zijlstra on Wed, 20 Sep 2006 18:27:04 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 11:14 -0700, Rohit Seth wrote:
> On Wed, 2006-09-20 at 20:06 +0200, Peter Zijlstra wrote:
> > On Wed, 2006-09-20 at 10:52 -0700, Christoph Lameter wrote:
> > > On Wed, 20 Sep 2006, Rohit Seth wrote:
> > >
> > > > Right now the memory handler in this container subsystem is written in
> > > > such a way that when existing kernel reclaimer kicks in, it will first
> > > > operate on those (container with pages over the limit) pages first. But
> > > > in general I like the notion of containerizing the whole reclaim code.
> > >
> > > Which comes naturally with cpusets.
> >
> > How are shared mappings dealt with, are pages charged to the set that
> > first faults them in?
> >
>
> For anonymous pages (simpler case), they get charged to the faulting
> task's container.
>
> For filesystem pages (could be shared across tasks running different
> containers): Every time a new file mapping is created, it is bound to a
> container of the process creating that mapping. All subsequent pages
> belonging to this mapping will belong to this container, irrespective of
> different tasks running in different containers accessing these pages.
> Currently, I've not implemented a mechanism to allow a file to be
> specifically moved into or out of container. But when that gets
> implemented then all pages belonging to a mapping will also move out of
> container (or into a new container).

Yes, I read that in your patches, I was wondering how the cpuset
approach would handle this.

Neither are really satisfactory for shared mappings.

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Chandra Seetharaman on Wed, 20 Sep 2006 18:34:00 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 09:25 -0700, Christoph Lameter wrote:
> On Tue, 19 Sep 2006, Rohit Seth wrote:
>
> > For example, a user can run a batch job like backup inside containers.
> > This job if run unconstrained could step over most of the memory present
> > in system thus impacting other workloads running on the system at that
> > time. But when the same job is run inside containers then the backup
> > job is run within container limits.
>
> I just saw this for the first time since linux-mm was not cced. We have
> discussed a similar mechanism on linux-mm.
>
> We already have such a functionality in the kernel its called a cpuset. A

Christoph,

There had been multiple discussions in the past (as recent as Aug 18, 2006), where we (Paul and CKRM/RG folks) have concluded that cpuset and resource management are orthogonal features.

cpuset provides "resource isolation", and what we, the resource management guys want is work-conserving resource control.

cpuset partitions resource and hence the resource that are assigned to a node is not available for other cpuset, which is not good for "resource management".

chandra
PS:
Aug 18 link: http://marc.theaimsgroup.com/?l=linux-kernel&m=115593114408336&w=2

Feb 2005 thread: http://marc.theaimsgroup.com/?l=ckrm-tech&m=110790400330617&w=2
> container could be created simply by creating a fake node that then
> allows constraining applications to this node. We already track the
> types of pages per node. The statistics you want are already existing.
> See /proc/zoneinfo and /sys/devices/system/node/node*/*.
>
> > We use the term container to indicate a structure against which we track
> > and charge utilization of system resources like memory, tasks etc for a
> > workload. Containers will allow system admins to customize the
> > underlying platform for different applications based on their
> > performance and HW resource utilization needs. Containers contain
> > enough infrastructure to allow optimal resource utilization without
> > bogging down rest of the kernel. A system admin should be able to
> > create, manage and free containers easily.
>

> Right thats what cpusets do and it has been working fine for years. Maybe
> Paul can help you if you find anything missing in the existing means to
> control resources.
>
>
--

-- ----------
Chandra Seetharaman | Be careful what you choose....
- sekharan@us.ibm.com | .....you may get it.
-- ----------

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Peter Zijlstra on Wed, 20 Sep 2006 18:37:42 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 10:50 -0700, Rohit Seth wrote:
> On Thu, 2006-09-21 at 03:00 +1000, Nick Piggin wrote:
> > (this time to the lists as well)
> >
> > Peter Zijlstra wrote:
> >
> > > I'd much rather containterize the whole reclaim code, which should not
> > > be too hard since he already adds a container pointer to struct page.
> >
> >
>
> Right now the memory handler in this container subsystem is written in
> such a way that when existing kernel reclaimer kicks in, it will first
> operate on those (container with pages over the limit) pages first. But
> in general I like the notion of containerizing the whole reclaim code.

Patch 5/5 seems to have a horrid deactivation scheme.

> > > I still have to reread what Rohit does for file backed pages, that gave
> > > my head a spin.
>
> Please let me know if there is any specific part that isn't making much
> sense.

Well, the whole over the limit handler is quite painfull, having taken a
second reading it isn't all that complex after all, just odd.

You just start invalidating whole files for file backed pages. Granted,
this will get you below the threshold. but you might just have destroyed
your working set.

Pretty much the same for you anonymous memory handler, you scan through
the pages in linear fashion and demote the first that you encounter.

Both things pretty thoroughly destroy the existing kernel reclaim.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 18:38:08 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 20:27 +0200, Peter Zijlstra wrote:

> Yes, I read that in your patches, I was wondering how the cpuset
> approach would handle this.
>
> Neither are really satisfactory for shared mappings.
>

In which way? We could have the per container flag indicating whether
to charge this container for shared mapping that it initiates or to the
container where mapping belongs...or is there something different that
you are referring.

-rohit

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Paul Menage on Wed, 20 Sep 2006 18:43:02 GMT
View Forum Message <> Reply to Message

On 9/20/06, Chandra Seetharaman <sekharan@us.ibm.com> wrote:
> > We already have such a functionality in the kernel its called a cpuset. A
>
> Christoph,
>
> There had been multiple discussions in the past (as recent as Aug 18,
> 2006), where we (Paul and CKRM/RG folks) have concluded that cpuset and
> resource management are orthogonal features.
>
> cpuset provides "resource isolation", and what we, the resource

> management guys want is work-conserving resource control.

CPUset provides two things:

- a generic process container abstraction

- "resource controllers" for CPU masks and memory nodes.

Rather than adding a new process container abstraction, wouldn't it
make more sense to change cpuset to make it more extensible (more
separation between resource controllers), possibly rename it to
"containers", and let the various resource controllers fight it out
(e.g. zone/node-based memory controller vs multiple LRU controller,
CPU masks vs a properly QoS-based CPU scheduler, etc)

Or more specifically, what would need to be added to cpusets to make
it possible to bolt the CKRM/RG resource controllers on to it?

Paul

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 18:57:40 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 20:37 +0200, Peter Zijlstra wrote:
> On Wed, 2006-09-20 at 10:50 -0700, Rohit Seth wrote:
> > On Thu, 2006-09-21 at 03:00 +1000, Nick Piggin wrote:
> > > (this time to the lists as well)
> > >
> > > Peter Zijlstra wrote:
> > >
> > > > I'd much rather containterize the whole reclaim code, which should not
> > > > be too hard since he already adds a container pointer to struct page.
> > >
> > >
> >
> > Right now the memory handler in this container subsystem is written in
> > such a way that when existing kernel reclaimer kicks in, it will first
> > operate on those (container with pages over the limit) pages first. But
> > in general I like the notion of containerizing the whole reclaim code.
>
> Patch 5/5 seems to have a horrid deactivation scheme.
>
> > > > I still have to reread what Rohit does for file backed pages, that gave
> > > > my head a spin.
> >
> > Please let me know if there is any specific part that isn't making much

> > sense.
>
> Well, the whole over the limit handler is quite painfull, having taken a
> second reading it isn't all that complex after all, just odd.
>

It is very basic right now.

> You just start invalidating whole files for file backed pages. Granted,
> this will get you below the threshold. but you might just have destroyed
> your working set.
>

When a container gone over the limit then it is okay to penalize it. I agree that I'm not making an attempt to maintain the current working set. Any suggestions that I can incorporate to improve this algorithm will be very appreciated.

> Pretty much the same for you anonymous memory handler, you scan through
> the pages in linear fashion and demote the first that you encounter.
>
> Both things pretty thoroughly destroy the existing kernel reclaim.
>

I agree that with in a container I need to do add more smarts to (for example) not do a linear search. Simple additions like last task or last mapping visited could be useful. And I definitely want to improve on that.

Though it should not destroy the existing kernel reclaim. Pages belonging to over the limit container should be the first ones to either get flushed out to FS or swapped if necessary. (Means that is the cost that you will have to pay if you, for example, want to container your tar to 100MB memory foot print).

-rohit

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Chandra Seetharaman on Wed, 20 Sep 2006 19:09:14 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 09:25 -0700, Christoph Lameter wrote:

For some reason the email i sent about 30 mins back didn't make it...
her is a resend.

> On Tue, 19 Sep 2006, Rohit Seth wrote:

>
> > For example, a user can run a batch job like backup inside containers.
> > This job if run unconstrained could step over most of the memory present
> > in system thus impacting other workloads running on the system at that
> > time. But when the same job is run inside containers then the backup
> > job is run within container limits.
>
> I just saw this for the first time since linux-mm was not cced. We have
> discussed a similar mechanism on linux-mm.
>
> We already have such a functionality in the kernel its called a cpuset. A

Christoph,

There had been multiple discussions in the past (as recent as Aug 18, 2006), where we (Paul and CKRM/RG folks) have concluded that cpuset and resource management are orthogonal features.

cpuset provides "resource isolation", and what we, the resource management guys want is work-conserving resource control.

cpuset partitions resource and hence the resource that are assigned to a node is not available for other cpuset, which is not good for "resource management".

chandra
PS:
Aug 18 link: http://marc.theaimsgroup.com/?l=linux-kernel&m=115593114408336&w=2

Feb 2005 thread: http://marc.theaimsgroup.com/?l=ckrm-tech&m=110790400330617&w=2

> container could be created simply by creating a fake node that then
> allows constraining applications to this node. We already track the
> types of pages per node. The statistics you want are already existing.
> See /proc/zoneinfo and /sys/devices/system/node/node*/*.
>
> > We use the term container to indicate a structure against which we track
> > and charge utilization of system resources like memory, tasks etc for a
> > workload. Containers will allow system admins to customize the
> > underlying platform for different applications based on their
> > performance and HW resource utilization needs. Containers contain
> > enough infrastructure to allow optimal resource utilization without
> > bogging down rest of the kernel. A system admin should be able to
> > create, manage and free containers easily.
>
> Right thats what cpusets do and it has been working fine for years. Maybe

> Paul can help you if you find anything missing in the existing means to
> control resources.
>
> --- -------------
> Take Surveys. Earn Cash. Influence the Future of IT
> Join SourceForge.net's Techsay panel and you'll get the chance to share your
> opinions on IT & business topics through brief surveys -- and earn cash
> http://www.techsay.com/default.php?page=join.php&p=sourc eforge&CID=DEVDEV
> _____
> ckrm-tech mailing list
> https://lists.sourceforge.net/lists/listinfo/ckrm-tech
--

-- ----------
Chandra Seetharaman | Be careful what you choose....
- sekharan@us.ibm.com | .....you may get it.
-- ----------

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Wed, 20 Sep 2006 19:48:13 GMT
View Forum Message <> Reply to Message

Peter wrote:
> > Which comes naturally with cpusets.
>
> How are shared mappings dealt with, are pages charged to the set that
> first faults them in?

Cpusets does not attempt to manage how much memory a task can allocate,
but where it can allocate it. If a task can find an existing page to
share, and avoid the allocation, then it entirely avoids dealing with
cpusets in that case.

Cpusets pays no attention to how often a page is shared. It controls
which tasks can allocate a given free page, based on the node on which
that page resides. If that node is allowed in a tasks 'nodemask_t
mems_allowed' (a task struct field), then the task can allocate
that page, so far as cpusets is concerned.

Cpusets does not care who links to a page, once it is allocated.

Every page is assigned to one specific node, and may only be allocated
by tasks allowed to allocate from that node.

These cpusets can overlap - which so far as memory goes, roughly means
that the various mems_allowed nodemask_t's of different tasks can overlap.

Here's an oddball example configuration that might make this easier to think about.

> Let's say we have a modest sized NUMA system with an extra bank of memory added, in addition to the per-node memory. Let's say the extra bank is a huge pile of cheaper (slower) memory, off a slower bus.
>
> Normal sized tasks running on one or more of the NUMA nodes just get to fight for the CPUs and memory on those nodes allowed them.
>
> Let's say an occassional big memory job is to be allowed to use some of the extra cheap memory, and we use the idea of Andrew and others to split that memory into fake nodes to manage the portion of memory available to specified tasks.
>
> Then one of these big jobs could be in a cpuset that let it use one or more of the CPUs and memory on the node it ran on, plus some number of the fake nodes on the extra cheap memory.
>
> Other jobs could be allowed, using cpusets, to use any combination of the same or overlapping CPUs or nodes, and/or other disjoint CPUs or nodes, fake or real.

Another example, restating some of the above.

> If say some application happened to fault in a libc.so page, it would be required to place that page on one of the nodes allowed to it. If an other application comes along later and ends up wanting shared references to that same page, it could certainly do so, regardless of its cpuset settings. It would not be allocating a new page for this, so would not encounter the cpuset constraints on where it could allocate such a page.

--

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

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 19:48:41 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Peter Zijlstra wrote:

> > Which comes naturally with cpusets.
>

> How are shared mappings dealt with, are pages charged to the set that
> first faults them in?

They are charged to the node from which they were allocated. If the
process is restricted to the node (container) then all pages allocated
are are charged to the container regardless if they are shared or not.

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Paul Menage on Wed, 20 Sep 2006 19:51:00 GMT
View Forum Message <> Reply to Message

On 9/20/06, Christoph Lameter <clameter@sgi.com> wrote:
> On Wed, 20 Sep 2006, Peter Zijlstra wrote:
>
> > > Which comes naturally with cpusets.
> >
> > How are shared mappings dealt with, are pages charged to the set that
> > first faults them in?
>
> They are charged to the node from which they were allocated. If the
> process is restricted to the node (container) then all pages allocated
> are are charged to the container regardless if they are shared or not.
>

Or you could use the per-vma mempolicy support to bind a large data
file to a particular node, and track shared file usage that way.

Paul

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 19:51:13 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Rohit Seth wrote:

> I thought the fake NUMA support still does not work on x86_64 baseline
> kernel. Though Paul and Andrew have patches to make it work.

Read linux-mm. There is work in progress.

> > This is commonly discussed under the subject of memory hotplug.
> So now we depend on getting memory hot-plug to work for faking up these
> nodes ...for the memory that is already present in the system. It just
> does not sound logical.

It is logical since nodes are containers of memory today and we have
established VM functionality to deal with these containers. If you read
the latest linx-mm then you will find that this was tested and works fine.

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 19:52:56 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Chandra Seetharaman wrote:

> cpuset partitions resource and hence the resource that are assigned to a
> node is not available for other cpuset, which is not good for "resource
> management".

cpusets can have one node in multiple cpusets.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Wed, 20 Sep 2006 20:06:56 GMT
View Forum Message <> Reply to Message

Seth wrote:
> I thought the fake NUMA support still does not work on x86_64 baseline
> kernel. Though Paul and Andrew have patches to make it work.

It works. Having long zonelists where one expects to have to scan a
long way down the list has a performance glitch, in the
get_page_from_freelist() code sucks. We don't want to be doing a
linear scan of a long list on this code path.

The cpuset_zone_allowed() routine happens to be the most obvious canary
in this linear scan loop (google 'canary in the mine shaft' for the
idiom), so shows up the problem first.

We don't have patches yet to fix this (well, we might, I still haven't
digested the last couple days worth of postings.) But we are persuing
Andrew's suggestion to cache the zone that we found memory on last time
around, so as to dramatically reduce the chance we have to rescan the
entire dang zonelist every time through this code.

Initially these zonelists had been designed to handle the various
kinds of dma, main and upper memory on common PC architectures, then
they were (ab)used to handle multiple Non-Uniform Memory Nodes (NUMA)
on bigger boxen. So it is not entirely surprising that we hit a
performance speed bump when further (ab)using them to handle multiple
Uniform sub-nodes as part of a memory containerization effort. Each

different kind of use hits these algorithms and data structures
differently.

It seems pretty clear by now that we will be able to pave over this
speed bump without doing any major reconstruction.

--

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

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Wed, 20 Sep 2006 22:27:53 GMT
View Forum Message <> Reply to Message

Christoph, responding to Nick:
> > Look at what the patches do. These are not only for hard partitioning
> > of memory per container but also those that share memory (eg. you might
> > want each to share 100MB of memory, up to a max of 80MB for an individual
> > container).
>
> So far I have not been able to find the hooks to the VM. The sharing
> would also work with nodes. Just create a couple of nodes with the sizes you
> want and then put the node with the shared memory into the cpusets for the
> apps sharing them.

Cpusets certainly allows for sharing - in the sense that multiple
tasks can be each be allowed to allocate from the same node (fake
or real.)

However, this is not sharing quite in the sense that Nick describes it.

In cpuset sharing, it is predetermined which pages are allowed to be
allocated by which tasks. Not "how many" pages, but "just which" pages.

Let's say we carve this 100 MB's up into 5 cpusets, of 20 MBs each, and
allow each of our many tasks to allocate from some specified 4 of these
5 cpusets. Then, even if some of those 100 MB's were still free, and
if a task was well below its allowed 80 MB's, the task might still not
be able to use that free memory, if that free memory happened to be in
whatever was the 5th cpuset that it was not allowed to use.

Seth:

> Could your container proposal handle the above example, and let that
> task have some of that memory, up to 80 MB's if available, but not
> more, regardless of what node the free memory was on?

I presume so.

Another example that highlights this difference - airline overbooking.
If an airline has to preassign every seat, it can't overbook, short of
putting two passengers in the same seat and hoping one is a no show,
which is pretty cut throat. If an airline is willing to bet that
seldom more than 90% of the ticketed passengers will show up, and it
doesn't preassign all seats, it can wait until flight time, see who
shows up, and hand out the seats then. It can preassign some seats,
but it needs some passengers showing up unassigned, free to take what's
left over.

Cpusets preassigns which nodes are allowed a task. If not all the
pages on a node are allocated by one of the tasks it is preassigned to,
those pages "fly empty" -- remain unallocated. This happens regardless
of how overbooked is the memory on other nodes.

If you just want to avoid fisticuffs at the gate between overbooked
passengers, cpusets are enough. If you further want to maximize utilization,
then you need the capacity management of resource groups, or some such.

> > The nodes+cpusets stuff doesn't seem to help with that because you
> > with that because you fundamentally need to track pages on a per
> > container basis otherwise you don't know who's got what.
>
> Hmmm... That gets into issues of knowing how many pages are in use by an
> application and that is fundamentally difficult to do due to pages being
> shared between processes.

Fundamentally difficult or not, it seems to be required for what Nick
describes, and for sure cpusets doesn't do it (track memory usage per
container.)

> > Now if, in practice, it turns out that nobody really needed these
> > features then of course I would prefer the cpuset+nodes approach. My
> > point is that I am not in a position to know who wants what, so I
> > hope people will come out and discuss some of these issues.

I don't know either ;).

--

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

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Wed, 20 Sep 2006 22:51:36 GMT
View Forum Message <> Reply to Message

Seth wrote:
> But am not sure
> if this number of nodes can change dynamically on the running machine or
> a reboot is required to change the number of nodes.

The current numa=fake=N kernel command line option is just boottime,
and just x86_64.

I presume we'd have to remove these two constraints for this to be
generally usable to containerize memory.

We also, in my current opinion, need to fix up the node_distance
between such fake numa sibling nodes, to correctly reflect that they
are on the same real node (LOCAL_DISTANCE).

And some non-trivial, arch-specific, zonelist sorting and reconstruction
work will be needed.

And an API devised for the above mentioned dynamic changing.

And this will push on the memory hotplug/unplug technology.

All in all, it could avoid anything more than trivial changes to the
existing memory allocation code hot paths. But the infrastructure
needed for managing this mechanism needs some non-trivial work.

> Though when you want to have in access of 100 containers then the cpuset
> function starts popping up on the oprofile chart very aggressively.

As the linux-mm discussion last weekend examined in detail, we can
eliminate this performance speed bump, probably by caching the
last zone on which we found some memory. The linear search that was
implicit in __alloc_pages()'s use of zonelists for many years finally
become explicit with this new usage pattern.

> Containers also provide a mechanism to move files to containers. Any
> further references to this file come from the same container rather than
> the container which is bringing in a new page.

I haven't read these patches enough to quite make sense of this, but I
suspect that this is not a distinction between cpusets and these
containers, for the basic reason that cpusets doesn't need to 'move'
a file's references because it has no clue what such are.

> In future there will be more handlers like CPU and disk that can be
> easily embeded into this container infrastructure.

This may be a deciding point.

--

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

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Wed, 20 Sep 2006 22:58:15 GMT
View Forum Message <> Reply to Message

Seth wrote:
> So now we depend on getting memory hot-plug to work for faking up these
> nodes ...for the memory that is already present in the system. It just
> does not sound logical.

It's logical to me. Part of memory hotplug is adding physial memory,
which is not an issue here. Part of it is adding another logical
memory node (turning on another bit in node_online_map) and fixing up
any code that thought a systems memory nodes were baked in at boottime.
Perhaps the hardest part is the memory hot-un-plug, which would become
more urgently needed with such use of fake numa nodes. The assumption
that memory doesn't just up and vanish is non-trivial to remove from
the kernel. A useful memory containerization should (IMHO) allow for
both adding and removing such containers.

--

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

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 22:59:03 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Paul Jackson wrote:

> > Hmmm... That gets into issues of knowing how many pages are in use by an
> > application and that is fundamentally difficult to do due to pages being
> > shared between processes.

>
> Fundamentally difficult or not, it seems to be required for what Nick
> describes, and for sure cpusets doesn't do it (track memory usage per
> container.)

We have the memory usage on a node. So in that sense we track memory usage. We also have VM counters for that node or nodes that describe resource usage.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 23:01:09 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Paul Jackson wrote:

> All in all, it could avoid anything more than trivial changes to the
> existing memory allocation code hot paths. But the infrastructure
> needed for managing this mechanism needs some non-trivial work.

This is material we have to anyways for hotplug support. Adding a real node or a virtual node whatever it is fundamentally the same process that requires a regeneration of the zonelists in the system.

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 23:02:25 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Paul Jackson wrote:

> the kernel. A useful memory containerization should (IMHO) allow for
> both adding and removing such containers.

How does the containers implementation under discussion behave if a process is part of a container and the container is removed?

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Wed, 20 Sep 2006 23:18:31 GMT
View Forum Message <> Reply to Message

Chistroph, responding to Alan:
> > I'm also not clear how you handle shared pages correctly under the fake
> > node system, can you perhaps explain that further how this works for say
> > a single apache/php/glibc shared page set across 5000 containers each a
> > web site.

>
> Cpusets can share nodes. I am not sure what the problem would be? Paul may
> be able to give you more details.

Cpusets share pre-assigned nodes, but not anonymous proportions of the
total system memory.

So sharing an apache/php/glibc page set across 5000 containers using
cpusets would be awkward. Unless I'm missing something, you'd have to
prepage in that page set, from some task allowed that many pages in
its own cpuset, then you'd run each of the 5000 web servers in smaller
cpusets that allowed space for the remainder of whatever that web
server was provisioned, not counting the shared pages. The shared pages
wouldn't count, because cpusets doesn't ding you for using a page that
is already in memory -- it just keeps you from allocating fresh pages
on certain nodes. When it came time to do rolling upgrades to new
versions of the software, and add a marketing driven list of 57
additional applications that the customers could use to build their
website, this could become an official nightmare.

Overbooking (selling say 10 Mbs of memory for each server, even though
there is less than 5000 * 10 Mb total RAM in the system) would also be
awkward. One could simulate with overlapping sets of fake numa nodes,
as I described in an earlier post today (the one that gave each task
some four of the five 20 MB fake cpusets.) But there would still be
false resource conflicts, and the (ab)use of the cpuset apparatus for
this seems unintuitive, in my opinion.

I imagine that a web site supporting 5000 web servers would be very
interested in overbooking working well. I'm sure the $7.99/month
cheap as dirt virtual web servers of which I am a customer overbook.

--

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

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 23:22:22 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 15:51 -0700, Paul Jackson wrote:
> Seth wrote:
> > But am not sure
> > if this number of nodes can change dynamically on the running machine or
> > a reboot is required to change the number of nodes.
>

> The current numa=fake=N kernel command line option is just boottime,
> and just x86_64.
>

Ah okay.

> I presume we'd have to remove these two constraints for this to be
> generally usable to containerize memory.
>

Right.

> We also, in my current opinion, need to fix up the node_distance
> between such fake numa sibling nodes, to correctly reflect that they
> are on the same real node (LOCAL_DISTANCE).
>
> And some non-trivial, arch-specific, zonelist sorting and reconstruction
> work will be needed.
>
> And an API devised for the above mentioned dynamic changing.
>
> And this will push on the memory hotplug/unplug technology.
>

Yes, if we use the existing notion of nodes for other purposes then you
have captured the right set of changes that will be needed to make that
happen. Such changes are not required for container patches as such.

> All in all, it could avoid anything more than trivial changes to the
> existing memory allocation code hot paths. But the infrastructure
> needed for managing this mechanism needs some non-trivial work.
>
>
> > Though when you want to have in access of 100 containers then the cpuset
> > function starts popping up on the oprofile chart very aggressively.
>
> As the linux-mm discussion last weekend examined in detail, we can
> eliminate this performance speed bump, probably by caching the
> last zone on which we found some memory. The linear search that was
> implicit in __alloc_pages()'s use of zonelists for many years finally
> become explicit with this new usage pattern.
>

Okay.

>
> > Containers also provide a mechanism to move files to containers. Any
> > further references to this file come from the same container rather than

> > the container which is bringing in a new page.
>
> I haven't read these patches enough to quite make sense of this, but I
> suspect that this is not a distinction between cpusets and these
> containers, for the basic reason that cpusets doesn't need to 'move'
> a file's references because it has no clue what such are.
>

But container support will allow the certain files pages to come from
the same container irrespective of who is using them. Something useful
for shared libs etc.

-rohit

>
> > In future there will be more handlers like CPU and disk that can be
> > easily embeded into this container infrastructure.
>
> This may be a deciding point.
>

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 23:26:48 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 15:58 -0700, Paul Jackson wrote:
> Seth wrote:
> > So now we depend on getting memory hot-plug to work for faking up these
> > nodes ...for the memory that is already present in the system. It just
> > does not sound logical.
>
> It's logical to me. Part of memory hotplug is adding physial memory,
> which is not an issue here. Part of it is adding another logical
> memory node (turning on another bit in node_online_map) and fixing up
> any code that thought a systems memory nodes were baked in at boottime.
> Perhaps the hardest part is the memory hot-un-plug, which would become
> more urgently needed with such use of fake numa nodes. The assumption
> that memory doesn't just up and vanish is non-trivial to remove from
> the kernel. A useful memory containerization should (IMHO) allow for
> both adding and removing such containers.
>

Absolutely. Since these containers are not (hard) partitioning the
memory in any way so it is easy to change the limits (effectively
reducing and increasing the memory limits for tasks belonging to
containers). As you said, memory hot-un-plug is important and it is
non-trivial amount of work.

-rohit

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Wed, 20 Sep 2006 23:29:45 GMT
View Forum Message <> Reply to Message

Alan replying to Christoph:
> > Cpusets can share nodes. I am not sure what the problem would be? Paul may
> > be able to give you more details.
>
> If it can do it in a human understandable way, configured at runtime
> with dynamic sharing, overcommit and reconfiguration of sizes then
> great. Lets see what Paul has to say.

Unless I'm missing something (a frequent occurrence) such a use of
cpusets looses on the understandable, is hobbled on the overcommit, and
has to make do with a somewhat oddly limited and not trivial to
configure approximation of the dynamic sharing. And the
reconfiguration would seem to be a great exercise of memory hotunplug
(echos of the original motivation for fake numa - exercising cpusets ;).

Not exactly passing with flying colors ;).

--

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

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 23:31:22 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Rohit Seth wrote:

> Absolutely. Since these containers are not (hard) partitioning the
> memory in any way so it is easy to change the limits (effectively
> reducing and increasing the memory limits for tasks belonging to
> containers). As you said, memory hot-un-plug is important and it is
> non-trivial amount of work.

Maybe the hotplug guys want to contribute to the discussion?

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 23:33:51 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 16:02 -0700, Christoph Lameter wrote:
> On Wed, 20 Sep 2006, Paul Jackson wrote:
>
> > the kernel. A useful memory containerization should (IMHO) allow for
> > both adding and removing such containers.
>
> How does the containers implementation under discussion behave if a
> process is part of a container and the container is removed?
>

It first removes all the tasks belonging to this container (which means
resetting the container pointers in task_struct and then per page
container pointer belonging to anonymous pages). It then clears the
container pointers in the mapping structure and also in the pages
belonging to these files.

-rohit

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 23:36:25 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006, Rohit Seth wrote:

> > How does the containers implementation under discussion behave if a
> > process is part of a container and the container is removed?
> It first removes all the tasks belonging to this container (which means
> resetting the container pointers in task_struct and then per page
> container pointer belonging to anonymous pages). It then clears the
> container pointers in the mapping structure and also in the pages
> belonging to these files.

So the application continues to run unharmed?

Could we equip containers with restrictions on processors and nodes for
NUMA?

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 20 Sep 2006 23:39:29 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 16:36 -0700, Christoph Lameter wrote:

> On Wed, 20 Sep 2006, Rohit Seth wrote:
>
> > > How does the containers implementation under discussion behave if a
> > > process is part of a container and the container is removed?
> > It first removes all the tasks belonging to this container (which means
> > resetting the container pointers in task_struct and then per page
> > container pointer belonging to anonymous pages). It then clears the
> > container pointers in the mapping structure and also in the pages
> > belonging to these files.
>
> So the application continues to run unharmed?
>

It will hit a one time penalty of getting those pointers reset, but
besides that it will continue to run fine.

> Could we equip containers with restrictions on processors and nodes for
> NUMA?
>

Yes. That is something we will have to do (I think part of CPU
handler-TBD).

-rohit

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Wed, 20 Sep 2006 23:45:52 GMT
View Forum Message <> Reply to Message

Seth wrote:
> But container support will allow the certain files pages to come from
> the same container irrespective of who is using them. Something useful
> for shared libs etc.

Yes - that is useful for shared libs, and your container patch
apparently does that cleanly, while cpuset+fakenuma containers can
only provide a compromise kludge.

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

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Christoph Lameter on Wed, 20 Sep 2006 23:51:45 GMT

On Wed, 20 Sep 2006, Rohit Seth wrote:

> > Could we equip containers with restrictions on processors and nodes for
> > NUMA?
> Yes. That is something we will have to do (I think part of CPU
> handler-TBD).

Paul: Will we still need cpusets if that is there?

Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Thu, 21 Sep 2006 00:05:44 GMT

> > > Could we equip containers with restrictions on processors and nodes for
> > > NUMA?
> > Yes. That is something we will have to do (I think part of CPU
> > handler-TBD).
>
> Paul: Will we still need cpusets if that is there?

Yes. There's quite a bit more to cpusets than just some form,
any form, of CPU and Memory restriction. I can't imagine that
Containers, in any form, are going to replicate that API.

--

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

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Paul Menage on Thu, 21 Sep 2006 00:09:22 GMT

On 9/20/06, Paul Jackson <pj@sgi.com> wrote:
>
> Yes. There's quite a bit more to cpusets than just some form,
> any form, of CPU and Memory restriction. I can't imagine that
> Containers, in any form, are going to replicate that API.
>

That would be one of the nice aspects of a generic process container
abstraction linked to different resource controllers - you wouldn't
need to replicate the cpuset support, you could use it in parallel
with other resource controllers. (So e.g. use the cpusets support to

pin a group of processes on to a given set of CPU/memory nodes, and
then use the CKRM/RG CPU and disk/IO controllers to limit resource
usage within those nodes)

Paul

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Chandra Seetharaman on Thu, 21 Sep 2006 00:31:55 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-20 at 12:52 -0700, Christoph Lameter wrote:
> On Wed, 20 Sep 2006, Chandra Seetharaman wrote:
>
> > cpuset partitions resource and hence the resource that are assigned to a
> > node is not available for other cpuset, which is not good for "resource
> > management".
>
> cpusets can have one node in multiple cpusets.

AFAICS, That doesn't help me in over committing resources.

--

Chandra Seetharaman | Be careful what you choose....
- sekharan@us.ibm.com | .....you may get it.

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Thu, 21 Sep 2006 00:36:38 GMT
View Forum Message <> Reply to Message

Chandra wrote:
> AFAICS, That doesn't help me in over committing resources.

I agree - I don't think cpusets plus fake numa ... handles over commit.
You might could hack up a cheap substitute, but it wouldn't do the job.

--

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

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Paul Menage on Thu, 21 Sep 2006 00:42:22 GMT
View Forum Message <> Reply to Message

On 9/20/06, Paul Jackson <pj@sgi.com> wrote:
> Chandra wrote:
> > AFAICS, That doesn't help me in over committing resources.
>
> I agree - I don't think cpusets plus fake numa ... handles over commit.
> You might could hack up a cheap substitute, but it wouldn't do the job.

I have some patches locally that basically let you give out a small
set of nodes initially to a cpuset, and if memory pressure in
try_to_free_pages() passes a specified threshold, automatically
allocate one of the parent cpuset's unused memory nodes to the child
cpuset, up to specified limit. It's a bit ugly, but lets you trade of
performance vs memory footprint on a per-job basis (when combined with
fake numa to give lots of small nodes).

Paul

Subject: Re: [Lhms-devel] [patch00/05]: Containers(V2)- Introduction
Posted by KAMEZAWA Hiroyuki on Thu, 21 Sep 2006 00:47:58 GMT
View Forum Message <> Reply to Message

On Wed, 20 Sep 2006 16:31:22 -0700 (PDT)
Christoph Lameter <clameter@sgi.com> wrote:

> On Wed, 20 Sep 2006, Rohit Seth wrote:
>
> > Absolutely. Since these containers are not (hard) partitioning the
> > memory in any way so it is easy to change the limits (effectively
> > reducing and increasing the memory limits for tasks belonging to
> > containers). As you said, memory hot-un-plug is important and it is
> > non-trivial amount of work.
>
> Maybe the hotplug guys want to contribute to the discussion?
>
Ah, I'm reading threads with interest.

I think this discussion is about using fake nodes ('struct pgdat')
to divide system's memory into some chunks. Your thought is that
for resizing/adding/removing fake pgdat, memory-hot-plug codes may be useful.
correct ?

Now, memory-hotplug manages all memory by 'section' and allows adding/(removing)
section to pgdat.

Does this section-size handling meet container people's requirement ?
And do we need freeing page when pgdat is removed ?

I think at least SPARSEMEM is useful for fake nodes because 'struct page'
are not tied to pgdat. (DISCONTIGMEM uses node_start_pfn. SPARSEMEM not.)

-Kame

Subject: Re: [Lhms-devel] [patch00/05]: Containers(V2)- Introduction
Posted by KAMEZAWA Hiroyuki on Thu, 21 Sep 2006 01:29:54 GMT
View Forum Message <> Reply to Message

Self-response..

On Thu, 21 Sep 2006 09:51:00 +0900
KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:

> On Wed, 20 Sep 2006 16:31:22 -0700 (PDT)
> Christoph Lameter <clameter@sgi.com> wrote:
>
> > On Wed, 20 Sep 2006, Rohit Seth wrote:
> >
> > > Absolutely. Since these containers are not (hard) partitioning the
> > > memory in any way so it is easy to change the limits (effectively
> > > reducing and increasing the memory limits for tasks belonging to
> > > containers). As you said, memory hot-un-plug is important and it is
> > > non-trivial amount of work.
> >
> > Maybe the hotplug guys want to contribute to the discussion?
> >
> Ah, I'm reading threads with interest.

I wonder it may not good to use pgdat for resource controlling.

For example

In following scenario,
==
(1). add <pid> > /mnt/configfs/containers/my_container/add_task
(2). <pid> does some work.
(3). echo <pid> > /mnt/configfs/containers/my_container/rm_task
(4). echo <pid> > /mnt/configfs/containers/my_container2/add_task
==
(if fake-pgdat/memory-hotplug is used)
The pages used by <pid> in (2) will be accounted in 'my_container' after (3).
Is this user's wrong use of system ?

-Kame

Subject: Re: [ckrm-tech] [Lhms-devel] [patch00/05]: Containers(V2)- Introduction
Posted by Paul Menage on Thu, 21 Sep 2006 01:36:01 GMT
View Forum Message <> Reply to Message

On 9/20/06, KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:
> For example
>
> In following scenario,
> ==
> (1). add <pid> > /mnt/configfs/containers/my_container/add_task
> (2). <pid> does some work.
> (3). echo <pid> > /mnt/configfs/containers/my_container/rm_task
> (4). echo <pid> > /mnt/configfs/containers/my_container2/add_task
> ==
> (if fake-pgdat/memory-hotplug is used)
> The pages used by <pid> in (2) will be accounted in 'my_container' after (3).
> Is this user's wrong use of system ?

Yes. You can't use memory node partitioning for file pages in this way unless you have strict controls over who can access the data sets in question, and are careful to prevent people from moving between containers. So it's not suitable for all uses of resource-isolating containers.

Who is to say that the pages allocated in (2) *should* be reaccounted to my_container2 after (3)? Some people might want that, other (including me) might not.

Paul

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Chandra Seetharaman on Wed, 27 Sep 2006 19:50:44 GMT
View Forum Message <> Reply to Message

Rohit,

I finally looked into your memory controller patches. Here are some of the issues I see:

(All points below are in the context of page limit of containers being hit and the new code starts freeing up pages)

1. LRU is ignored totally thereby thrashing the working set (as pointed

by Peter Zijlstra).
2. Frees up file pages first when hitting the page limit thereby making vm_swappiness ineffective.
3. Starts writing back pages when the # of file pages is close to the limit, thereby breaking the current writeback algorithm/logic.
4. MAPPED files are not counted against the page limit. why ?. This affects reclamation behavior and makes vm_swappiness ineffective.
5. Starts freeing up pages from the first task or the first file in the linked list. This logic unfairly penalizes the early members of the list.
6. Both active and inactive pages use physical pages. But, the controller only counts active pages and not inactive pages. why ?
7. Page limit is checked against the sum of (anon and file pages) in some places and against active pages at some other places. IMO, it should be always compared to the same value.

BTW, It will be easier to read/follow the patches if you separate them out as functionalities.

regards,

chandra

On Tue, 2006-09-19 at 19:16 -0700, Rohit Seth wrote:
> Containers:
>
> Commodity HW is becoming more powerful. This is giving opportunity to
> run different workloads on the same platform for better HW resource
> utilization. To run different workloads efficiently on the same
> platform, it is critical that we have a notion of limits for each
> workload in Linux kernel. Current cpuset feature in Linux kernel
> provides grouping of CPU and memory support to some extent (for NUMA
> machines).
>
> For example, a user can run a batch job like backup inside containers.
> This job if run unconstrained could step over most of the memory present
> in system thus impacting other workloads running on the system at that
> time. But when the same job is run inside containers then the backup
> job is run within container limits.
>
> We use the term container to indicate a structure against which we track
> and charge utilization of system resources like memory, tasks etc for a
> workload. Containers will allow system admins to customize the
> underlying platform for different applications based on their
> performance and HW resource utilization needs. Containers contain
> enough infrastructure to allow optimal resource utilization without
> bogging down rest of the kernel. A system admin should be able to
> create, manage and free containers easily.

>
> At the same time, changes in kernel are minimized so as this support can
> be easily integrated with mainline kernel.
>
> The user interface for containers is through configfs. Appropriate file
> system privileges are required to do operations on each container.
> Currently implemented container resources are automatically visible to
> user space through /configfs/container/<container_name> after a
> container is created.
>
> Signed-off-by: Rohit Seth <rohitseth@google.com>
>
> Diffstat for the patch set (against linux-2.6.18-rc6-mm2_:
>
> Documentation/containers.txt | 65 ++++
> fs/inode.c | 3
> include/linux/container.h | 167 ++++++++++
> include/linux/fs.h | 5
> include/linux/mm_inline.h | 4
> include/linux/mm_types.h | 4
> include/linux/sched.h | 6
> init/Kconfig | 8
> kernel/Makefile | 1
> kernel/container_configfs.c | 440 ++++++++++++++++++++++++++++++
> kernel/exit.c | 2
> kernel/fork.c | 9
> mm/Makefile | 2
> mm/container.c | 658 ++
> mm/container_mm.c | 512 ++++++++++++++++++++++++++++++++
> mm/filemap.c | 4
> mm/page_alloc.c | 3
> mm/rmap.c | 8
> mm/swap.c | 1
> mm/vmscan.c | 1
> 20 files changed, 1902 insertions(+), 1 deletion(-)
>
> Changes from version 1:
> Fixed the Documentation error
> Fixed the corruption in container task list
> Added the support for showing all the tasks belonging to a container
> through showtask attribute
> moved the Kconfig changes to init directory (from mm)
> Fixed the bug of unregistering container subsystem if we are not able to
> create workqueue
> Better support for handling limits for file pages. This now includes
> support for flushing and invalidating page cache pages.
> Minor other changes.
>

> ********** *****
> This patch set has basic container support that includes:
>
> - Create a container using mkdir command in configfs
>
> - Free a container using rmdir command
>
> - Dynamically adjust memory and task limits for container.
>
> - Add/Remove a task to container (given a pid)
>
> - Files are currently added as part of open from a task that already
> belongs to a container.
>
> - Keep track of active, anonymous, mapped and pagecache usage of
> container memory
>
> - Does not allow more than task_limit number of tasks to be created in
> the container.
>
> - Over the limit memory handler is called when number of pages (anon +
> pagecache) exceed the limit. It is also called when number of active
> pages exceed the page limit. Currently, this memory handler scans the
> mappings and tasks belonging to container (file and anonymous) and tries
> to deactivate pages. If the number of page cache pages is also high
> then it also invalidate mappings. The thought behind this scheme is, it
> is okay for containers to go over limit as long they run in degraded
> manner when they are over their limit. Also, if there is any memory
> pressure then pages belonging to over the limit container(s) become
> prime candidates for kernel reclaimer. Container mutex is also held
> during the time this handler is working its way through to prevent any
> further addition of resources (like tasks or mappings) to this
> container. Though it also blocks removal of same resources from the
> container for the same time. It is possible that over the limit page
> handler takes lot of time if memory pressure on a container is
> continuously very high. The limits, like how long a task should
> schedule out when it hits memory limit, is also on the lower side at
> present (particularly when it is memory hogger). But should be easy to
> change if need be.
>
> - Indicate the number of times the page limit and task limit is hit
>
> - Indicate the tasks (pids) belonging to container.
>
> Below is a one line description for patches that will follow:
>
> [patch01]: Documentation on how to use containers
> (Documentation/container.txt)

>
> [patch02]: Changes in the generic part of kernel code
>
> [patch03]: Container's interface with configfs
>
> [patch04]: Core container support
>
> [patch05]: Over the limit memory handler.
>
> TODO:
>
> - some code(like container_add_task) in mm/container.c should go
> elsewhere.
> - Support adding/removing a file name to container through configfs
> - /proc/pid/container to show the container id (or name)
> - More testing for memory controller. Currently it is possible that
> limits are exceeded. See if a call to reclaim can be easily integrated.
> - Kernel memory tracking (based on patches from BC)
> - Limit on user locked memory
> - Huge memory support
> - Stress testing with containers
> - One shot view of all containers
> - CKRM folks are interested in seeing all processes belonging to a
> container. Add the attribute show_tasks to container.
> - Add logic so that the sum of limits are not exceeding appropriate
> system requirements.
> - Extend it with other controllers (CPU and Disk I/O)
> - Add flags bits for supporting different actions (like in some cases
> provide a hard memory limit and in some cases it could be soft).
> - Capability to kill processes for the extreme cases.
> ...
>
> This is based on lot of discussions over last month or so. I hope this
> patch set is something that we can agree and more support can be added
> on top of this. Please provide feedback and add other extensions that
> are useful in the TODO list.
>
> Thanks,
> -rohit
>
>
>
>
>
> -- -------------
> Take Surveys. Earn Cash. Influence the Future of IT
> Join SourceForge.net's Techsay panel and you'll get the chance to share your
> opinions on IT & business topics through brief surveys -- and earn cash

> http://www.techsay.com/default.php?page=join.php&p=sourc eforge&CID=DEVDEV
> _____
> ckrm-tech mailing list
> https://lists.sourceforge.net/lists/listinfo/ckrm-tech
--

---------- ----------
Chandra Seetharaman | Be careful what you choose....
- sekharan@us.ibm.com | .....you may get it.
---------- ----------

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Wed, 27 Sep 2006 21:28:06 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-27 at 12:50 -0700, Chandra Seetharaman wrote:
> Rohit,
>
> I finally looked into your memory controller patches. Here are some of
> the issues I see:
> (All points below are in the context of page limit of containers being
> hit and the new code starts freeing up pages)
>
> 1. LRU is ignored totally thereby thrashing the working set (as pointed
> by Peter Zijlstra).

As the container goes over the limit, this algorithm deactivates some of the pages. I agree that the logic to find out the correct pages to deactivate needs to be improved. But the idea is that these pages go in front of inactive list so that if there is any memory pressure system wide then these pages can easily be reclaimed.

> 2. Frees up file pages first when hitting the page limit thereby making
> vm_swappiness ineffective.

Not sure if I understood this part correctly. But the choice when the container goes over its limit is between swap out some of the anonymous memory first or writeback some of the dirty file pages belonging to this container.

> 3. Starts writing back pages when the # of file pages is close to the
> limit, thereby breaking the current writeback algorithm/logic.

That is done so as to ensure processes belonging to container (Whose limit is hit) are the first ones getting penalized. For example, if you run a tar in a container with 100MB limit then the dirty file pages will be written back to disk when 100MB limit is hit). Though I will be

adding a HARD_LIMIT on page cache flag and the strict limit will be only maintained if this container flag is set.

> 4. MAPPED files are not counted against the page limit. why ?. This
> affects reclamation behavior and makes vm_swappiness ineffective.

num_mapped_pages only indicates how many page cache pages are mapped in user page tables. More of an accounting variable.

> 5. Starts freeing up pages from the first task or the first file in the
> linked list. This logic unfairly penalizes the early members of the
> list.

This is the part that I've to fix. Some per container variables that remembers the last values will help here.

> 6. Both active and inactive pages use physical pages. But, the
> controller only counts active pages and not inactive pages. why ?

The thought is, it is okay for containers to go over its limit as long as there is enough memory in the system. When there is any memory pressure then the inactive (+ dereferenced) pages get swapped out thus penalizing the container. I'm also thinking of having hard limit for anonymous pages beyond which the container will not be able to grow its anonymous pages.

> 7. Page limit is checked against the sum of (anon and file pages) in
> some places and against active pages at some other places. IMO, it
> should be always compared to the same value.
>
It is checked against sum of anon+file pages at the time when new pages is getting allocated. But as the reclaimer activate the pages, so it is also important to make sure the number of active pages is not going above its limit.

Thanks for your comments,
-rohit

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Chandra Seetharaman on Wed, 27 Sep 2006 22:24:52 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-27 at 14:28 -0700, Rohit Seth wrote:

Rohit,

For 1-4, I understand the rationale. But, your implementation deviates

from the current behavior of the VM subsystem which could affect the
ability of these patches getting into mainline.

IMO, the current behavior in terms of reclamation, LRU, vm_swappiness,
and writeback logic should be maintained.

> On Wed, 2006-09-27 at 12:50 -0700, Chandra Seetharaman wrote:
> > Rohit,
> >
> > I finally looked into your memory controller patches. Here are some of
> > the issues I see:
> > (All points below are in the context of page limit of containers being
> > hit and the new code starts freeing up pages)
> >
> > 1. LRU is ignored totally thereby thrashing the working set (as pointed
> >   by Peter Zijlstra).
>
> As the container goes over the limit, this algorithm deactivates some of
> the pages. I agree that the logic to find out the correct pages to
> deactivate needs to be improved. But the idea is that these pages go in
> front of inactive list so that if there is any memory pressure system
> wide then these pages can easily be reclaimed.
>
> > 2. Frees up file pages first when hitting the page limit thereby making
> >   vm_swappiness ineffective.
>
> Not sure if I understood this part correctly. But the choice when the
> container goes over its limit is between swap out some of the anonymous
> memory first or writeback some of the dirty file pages belonging to this
> container.
>
> > 3. Starts writing back pages when the # of file pages is close to the
> >   limit, thereby breaking the current writeback algorithm/logic.
>
> That is done so as to ensure processes belonging to container (Whose
> limit is hit) are the first ones getting penalized. For example, if you
> run a tar in a container with 100MB limit then the dirty file pages will
> be written back to disk when 100MB limit is hit). Though I will be
> adding a HARD_LIMIT on page cache flag and the strict limit will be only
> maintained if this container flag is set.
>
> > 4. MAPPED files are not counted against the page limit. why ?. This
> >   affects reclamation behavior and makes vm_swappiness ineffective.
>
> num_mapped_pages only indicates how many page cache pages are mapped in
> user page tables. More of an accounting variable.

But, # of mapped pages is used in the reclamation path logic. These set

of patches doesn't take them into account.

>
> > 5. Starts freeing up pages from the first task or the first file in the
> >   linked list. This logic unfairly penalizes the early members of the
> >   list.
>
> This is the part that I've to fix. Some per container variables that
> remembers the last values will help here.

Yes, that will help in fairness between the items in the list.

But, it will still suffer from (1) above, as we would have no idea of
the current working set (LRU) (within an item or among the items).

>
> > 6. Both active and inactive pages use physical pages. But, the
> >   controller only counts active pages and not inactive pages. why ?
>
> The thought is, it is okay for containers to go over its limit as long

Real number of "physical pages" used by the container is the sum of
active and inactive pages.

My question is, shouldn't that be used to check against page limit
instead of active pages alone ?

How do we describe "page limit" as (to the user) ?

> as there is enough memory in the system. When there is any memory
> pressure then the inactive (+ dereferenced) pages get swapped out thus
> penalizing the container. I'm also thinking of having hard limit for

Reclamation goes through active pages and page cache pages before it
gets into inactive pages. So, this may not work as you are explaining.

> anonymous pages beyond which the container will not be able to grow its
> anonymous pages.

You might break the current behavior (memory pressure must be very high
before these starts failing) if you are going to be strict about it.

>
> > 7. Page limit is checked against the sum of (anon and file pages) in
> >   some places and against active pages at some other places. IMO, it
> >   should be always compared to the same value.
> >
> It is checked against sum of anon+file pages at the time when new pages

why can't we check against active pages here ?

> is getting allocated. But as the reclaimer activate the pages, so it is
> also important to make sure the number of active pages is not going
> above its limit.

My point is that they won't be same (ever) and hence the check is
inconsistent.

Also, if it is this way, how do we describe the purpose of "page
limit" (for the user).

--

---------- ----------
Chandra Seetharaman | Be careful what you choose....
- sekharan@us.ibm.com | .....you may get it.
---------- ----------

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Balbir Singh on Thu, 28 Sep 2006 08:01:33 GMT
View Forum Message <> Reply to Message

Chandra Seetharaman wrote:
> On Wed, 2006-09-27 at 14:28 -0700, Rohit Seth wrote:
>
> Rohit,
>
> For 1-4, I understand the rationale. But, your implementation deviates
> from the current behavior of the VM subsystem which could affect the
> ability of these patches getting into mainline.
>
> IMO, the current behavior in terms of reclamation, LRU, vm_swappiness,
> and writeback logic should be maintained.
>

<snip>

Hi, Rohit,

I have been playing around with the containers patch. I finally got
around to reading the code.

1. Comments on reclaiming

You could try the following options to overcome some of the disadvantages of the current scheme.

(a) You could consider a reclaim path based on Dave Hansen's Challenged memory controller (see http://marc.theaimsgroup.com/?l=linux-mm&m=1155669825323 45&w=2).

(b) The other option is to do what the resource group memory controller does - build a per group LRU list of pages (active, inactive) and reclaim them using the existing code (by passing the correct container pointer, instead of the zone pointer). One disadvantage of this approach is that the global reclaim is impacted as the global LRU list is broken. At the expense of another list, we could maintain two lists, global LRU and container LRU lists. Depending on the context of the reclaim - (container over limit, memory pressure) we could update/manipulate both lists. This approach is definitely very expensive.

2. Comments on task migration support

(a) One of the issues I found while using the container code is that, one could add a task to a container say "a". "a" gets charged for the tasks usage, when the same task moves to a different container say "b", when the task exits, the credit goes to "b" and "a" remains indefinitely charged.

(b) For tasks addition and removal, I think it's probably better to move the entire process (thread group) rather than allow each individual thread to move across containers. Having threads belonging to the same process reside in different containers can be complex to handle, since they share the same VM. Do you have a scenario where the above condition would be useful?

--

Warm Regards,
Balbir Singh,
Linux Technology Center,
IBM Software Labs

PS: Chandra, I hope the details of the resource group memory controller are correct.

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Thu, 28 Sep 2006 18:12:19 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-27 at 15:24 -0700, Chandra Seetharaman wrote:
> On Wed, 2006-09-27 at 14:28 -0700, Rohit Seth wrote:

>
> Rohit,
>
> For 1-4, I understand the rationale. But, your implementation deviates
> from the current behavior of the VM subsystem which could affect the
> ability of these patches getting into mainline.
>

I agree that this implementation differs from existing VM subsystem.
But the key point here is, it puts the pages that should be reclaimed.
And this part needs further refining.

> IMO, the current behavior in terms of reclamation, LRU, vm_swappiness,
> and writeback logic should be maintained.
>

How? I don't want to duplicate the whole logic for containers.

> > On Wed, 2006-09-27 at 12:50 -0700, Chandra Seetharaman wrote:
> > > Rohit,
> > >
> > > I finally looked into your memory controller patches. Here are some of
> > > the issues I see:
> > > (All points below are in the context of page limit of containers being
> > > hit and the new code starts freeing up pages)
> > >
> > > 1. LRU is ignored totally thereby thrashing the working set (as pointed
> > >   by Peter Zijlstra).
> >
> > As the container goes over the limit, this algorithm deactivates some of
> > the pages. I agree that the logic to find out the correct pages to
> > deactivate needs to be improved. But the idea is that these pages go in
> > front of inactive list so that if there is any memory pressure system
> > wide then these pages can easily be reclaimed.
> >
> > > 2. Frees up file pages first when hitting the page limit thereby making
> > >   vm_swappiness ineffective.
> >
> > Not sure if I understood this part correctly. But the choice when the
> > container goes over its limit is between swap out some of the anonymous
> > memory first or writeback some of the dirty file pages belonging to this
> > container.
> >
> > > 3. Starts writing back pages when the # of file pages is close to the
> > >   limit, thereby breaking the current writeback algorithm/logic.
> >
> > That is done so as to ensure processes belonging to container (Whose
> > limit is hit) are the first ones getting penalized. For example, if you

> > run a tar in a container with 100MB limit then the dirty file pages will
> > be written back to disk when 100MB limit is hit). Though I will be
> > adding a HARD_LIMIT on page cache flag and the strict limit will be only
> > maintained if this container flag is set.
> >
> > > 4. MAPPED files are not counted against the page limit. why ?. This
> > > affects reclamation behavior and makes vm_swappiness ineffective.
> >
> > num_mapped_pages only indicates how many page cache pages are mapped in
> > user page tables. More of an accounting variable.
>
> But, # of mapped pages is used in the reclamation path logic. These set
> of patches doesn't take them into account.
>
> >
> > > 5. Starts freeing up pages from the first task or the first file in the
> > > linked list. This logic unfairly penalizes the early members of the
> > > list.
> >
> > This is the part that I've to fix. Some per container variables that
> > remembers the last values will help here.
>
> Yes, that will help in fairness between the items in the list.
>
> But, it will still suffer from (1) above, as we would have no idea of
> the current working set (LRU) (within an item or among the items).
>

Please let me know how do you propose to have another LRU for pages in
containers. Though I can add some heuristics.

> >
> > > 6. Both active and inactive pages use physical pages. But, the
> > > controller only counts active pages and not inactive pages. why ?
> >
> > The thought is, it is okay for containers to go over its limit as long
>
> Real number of "physical pages" used by the container is the sum of
> active and inactive pages.
>

>From the user pov, the real sum of pages that are used by container for
user land is anon + file. Now some times it is possible that there are
active pages that are neither in page cache nor in use as anon.

> My question is, shouldn't that be used to check against page limit
> instead of active pages alone ?

I can use active+inactive as the test. Sure. But I will have to also
still have a check to make sure that number of active pages themselves
is not bigger than page_limit.

> How do we describe "page limit" as (to the user) ?
>

Amount of memory below which no container throttling will happen. And
if the system is properly configured that it also ensures that this much
memory will always be there to user. If a container goes over this
limit then it will be throttled and it will suffer performance.

> > as there is enough memory in the system. When there is any memory
> > pressure then the inactive (+ dereferenced) pages get swapped out thus
> > penalizing the container. I'm also thinking of having hard limit for
>
> Reclamation goes through active pages and page cache pages before it
> gets into inactive pages. So, this may not work as you are explaining.

That is a good point. I'll have to make a check in reclaim so that when
the system is ready for swap or write back then containers are looked
first.

>
> > anonymous pages beyond which the container will not be able to grow its
> > anonymous pages.
>
> You might break the current behavior (memory pressure must be very high
> before these starts failing) if you are going to be strict about it.
>

That feature when implemented will be a container specific.

> >
> > > 7. Page limit is checked against the sum of (anon and file pages) in
> > >   some places and against active pages at some other places. IMO, it
> > >   should be always compared to the same value.
> > >
> > It is checked against sum of anon+file pages at the time when new pages
>
> why can't we check against active pages here ?
>
> > is getting allocated. But as the reclaimer activate the pages, so it is
> > also important to make sure the number of active pages is not going
> > above its limit.
>
> My point is that they won't be same (ever) and hence the check is
> inconsistent.

>

The check ensures
1- when a new page is getting added then the total sum of pages is
checked against the limit.
2- Number of active pages don't exceed the limit.

These two points combined together enforce the decision that once the
container goes over the limit, we scan the pages again to deactivate the
excess.

Thanks,
-rohit

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Thu, 28 Sep 2006 18:31:15 GMT
View Forum Message <> Reply to Message

On Thu, 2006-09-28 at 13:31 +0530, Balbir Singh wrote:
> Chandra Seetharaman wrote:
> > On Wed, 2006-09-27 at 14:28 -0700, Rohit Seth wrote:
> >
> > Rohit,
> >
> > For 1-4, I understand the rationale. But, your implementation deviates
> > from the current behavior of the VM subsystem which could affect the
> > ability of these patches getting into mainline.
> >
> > IMO, the current behavior in terms of reclamation, LRU, vm_swappiness,
> > and writeback logic should be maintained.
> >
>
> <snip>
>
> Hi, Rohit,
>
> I have been playing around with the containers patch. I finally got
> around to reading the code.
>
>
> 1. Comments on reclaiming
>
> You could try the following options to overcome some of the disadvantages of the
> current scheme.
>
> (a) You could consider a reclaim path based on Dave Hansen's Challenged memory
> controller (see http://marc.theaimsgroup.com/?l=linux-mm&m=1155669825323 45&w=2).

>

I will go through that. Did you get a chance to stress the system and
found any short comings that should be resolved.

> (b) The other option is to do what the resource group memory controller does -
> build a per group LRU list of pages (active, inactive) and reclaim
> them using the existing code (by passing the correct container pointer,
> instead of the zone pointer). One disadvantage of this approach is that
> the global reclaim is impacted as the global LRU list is broken. At the
> expense of another list, we could maintain two lists, global LRU and
> container LRU lists. Depending on the context of the reclaim - (container
> over limit, memory pressure) we could update/manipulate both lists.
> This approach is definitely very expensive.
>

Two LRUs is a nice idea. Though I don't think it will go too far. It
will involve adding another list pointers in the page structure. I
agree that the mem handler is not optimal at all but I don't want to
make it mimic kernel reclaimer at the same time.

> 2. Comments on task migration support
>
> (a) One of the issues I found while using the container code is that, one could
> add a task to a container say "a". "a" gets charged for the tasks usage,
> when the same task moves to a different container say "b", when the task
> exits, the credit goes to "b" and "a" remains indefinitely charged.
>
hmm, when the task is removed from "a" then "a" gets the credits for the
amount of anon memory that is used by the task. Or do you mean
something different.

> (b) For tasks addition and removal, I think it's probably better to move
> the entire process (thread group) rather than allow each individual thread
> to move across containers. Having threads belonging to the same process
> reside in different containers can be complex to handle, since they
> share the same VM. Do you have a scenario where the above condition
> would be useful?
>
>
I don't have a scenario where a task actually gets to move out of
container (except exit). That asynchronous removal of tasks has already
got the code very complicated for locking etc. But if you think moving
a thread group is useful then I will add that functionality.

Thanks,
-rohit

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Chandra Seetharaman on Thu, 28 Sep 2006 20:23:04 GMT
View Forum Message <> Reply to Message

On Thu, 2006-09-28 at 11:12 -0700, Rohit Seth wrote:
> On Wed, 2006-09-27 at 15:24 -0700, Chandra Seetharaman wrote:
> > On Wed, 2006-09-27 at 14:28 -0700, Rohit Seth wrote:
> >
> > Rohit,
> >
> > For 1-4, I understand the rationale. But, your implementation deviates
> > from the current behavior of the VM subsystem which could affect the
> > ability of these patches getting into mainline.
> >
>
> I agree that this implementation differs from existing VM subsystem.
> But the key point here is, it puts the pages that should be reclaimed.

But, you are putting the pages up for reclamation without any
consideration to the working set and system memory pressure.

> And this part needs further refining.
>
> > IMO, the current behavior in terms of reclamation, LRU, vm_swappiness,
> > and writeback logic should be maintained.
> >
>
> How? I don't want to duplicate the whole logic for containers.

We don't have to be duplicating the whole logic. Just make sure that the
existing mechanisms are aware of containers, if they exist.

<snip>

> >
> > But, it will still suffer from (1) above, as we would have no idea of
> > the current working set (LRU) (within an item or among the items).
> >
>
> Please let me know how do you propose to have another LRU for pages in
> containers. Though I can add some heuristics.

There are multiple ways as Balbir pointed in his email:
- reclamation per container (as in current RG implementation)
  (+ do a system wide reclaim when the system pressure is high)
- reclaim with the knowledge of containers that are over limit
  (Dave Hansen's patches + avoid overhead of combing the list)
- have two lists one for the system and one per container

>
> > >
> > > > 6. Both active and inactive pages use physical pages. But, the
> > > > controller only counts active pages and not inactive pages. why ?
> > >
> > > The thought is, it is okay for containers to go over its limit as long
> >
> > Real number of "physical pages" used by the container is the sum of
> > active and inactive pages.
> >
>
> >From the user pov, the real sum of pages that are used by container for
> user land is anon + file. Now some times it is possible that there are
> active pages that are neither in page cache nor in use as anon.
>
>
> > My question is, shouldn't that be used to check against page limit
> > instead of active pages alone ?
> I can use active+inactive as the test. Sure. But I will have to also
> still have a check to make sure that number of active pages themselves
> is not bigger than page_limit.
>
> > How do we describe "page limit" as (to the user) ?
> >
>
> Amount of memory below which no container throttling will happen. And

But, from the implementation one cannot clearly derive what we mean by
"memory" here (physical ?, file + anon ?; if we say physical, it is not
correct).

> if the system is properly configured that it also ensures that this much
> memory will always be there to user. If a container goes over this
> limit then it will be throttled and it will suffer performance.

But, the user's expectation would be that we would be throwing out pages
based on LRU (within that container). But this implementation doesn't
provide that behavior. It doesn't care about the working set.

Performance impact will be lesser if we consider the working set and
throw out pages based on LRU (within a container).

>
> > > as there is enough memory in the system. When there is any memory
> > > pressure then the inactive (+ dereferenced) pages get swapped out thus
> > > penalizing the container. I'm also thinking of having hard limit for
> >
> > Reclamation goes through active pages and page cache pages before it

> > gets into inactive pages. So, this may not work as you are explaining.
>
> That is a good point. I'll have to make a check in reclaim so that when
> the system is ready for swap or write back then containers are looked
> first.
>
> >
> > > anonymous pages beyond which the container will not be able to grow its
> > > anonymous pages.
> >
> > You might break the current behavior (memory pressure must be very high
> > before these starts failing) if you are going to be strict about it.
> >
>
> That feature when implemented will be a container specific.

My point is, even though it is container specific, the behavior (inside
a container) should be same as what a user sees at the system level now.

For example, consider a workload that is run on a 1G system now, and
user sees only occasional memory allocation failures and just a handful
of oom kills. When the workload is moved to a container with 1G,
failures the user see should be in the same order (and similar with
performance characteristics).

Do you agree that it will be the user's expectation ?

>
> > >
> > > > 7. Page limit is checked against the sum of (anon and file pages) in
> > > >   some places and against active pages at some other places. IMO, it
> > > >   should be always compared to the same value.
> > > >
> > > It is checked against sum of anon+file pages at the time when new pages
> >
> > why can't we check against active pages here ?
> >
> > > is getting allocated. But as the reclaimer activate the pages, so it is
> > > also important to make sure the number of active pages is not going
> > > above its limit.
> >
> > My point is that they won't be same (ever) and hence the check is
> > inconsistent.
> >
>
> The check ensures
> 1- when a new page is getting added then the total sum of pages is
> checked against the limit.

> 2- Number of active pages don't exceed the limit.
>
> These two points combined together enforce the decision that once the
> container goes over the limit, we scan the pages again to deactivate the
> excess.

Again, I understand the rationale. But it is not consistent.
>
> Thanks,
> -rohit
>
--

---------- ----------
Chandra Seetharaman | Be careful what you choose....
- sekharan@us.ibm.com | .....you may get it.
---------- ----------

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Thu, 28 Sep 2006 21:38:36 GMT
View Forum Message <> Reply to Message

On Thu, 2006-09-28 at 13:23 -0700, Chandra Seetharaman wrote:
> On Thu, 2006-09-28 at 11:12 -0700, Rohit Seth wrote:
> > On Wed, 2006-09-27 at 15:24 -0700, Chandra Seetharaman wrote:
> > > On Wed, 2006-09-27 at 14:28 -0700, Rohit Seth wrote:
> > >
> > > Rohit,
> > >
> > > For 1-4, I understand the rationale. But, your implementation deviates
> > > from the current behavior of the VM subsystem which could affect the
> > > ability of these patches getting into mainline.
> > >
> >
> > I agree that this implementation differs from existing VM subsystem.
> > But the key point here is, it puts the pages that should be reclaimed.
>
> But, you are putting the pages up for reclamation without any
> consideration to the working set and system memory pressure.
>

And I agree with you that some heuristics need to be put there to make
that algorithm go better.

> > And this part needs further refining.
> >
> > > IMO, the current behavior in terms of reclamation, LRU, vm_swappiness,

> > > and writeback logic should be maintained.
> > >
> >
> > How? I don't want to duplicate the whole logic for containers.
>
> We don't have to be duplicating the whole logic. Just make sure that the
> existing mechanisms are aware of containers, if they exist.
>

The next version is going to have hooks in kernel reclaim path for
containers. But that will still not make it close to what normal
reclaim path does for pages outside containers.

> <snip>
>
> > >
> > > But, it will still suffer from (1) above, as we would have no idea of
> > > the current working set (LRU) (within an item or among the items).
> > >
> >
> > Please let me know how do you propose to have another LRU for pages in
> > containers. Though I can add some heuristics.
>
> There are multiple ways as Balbir pointed in his email:
> - reclamation per container (as in current RG implementation)
>   (+ do a system wide reclaim when the system pressure is high)
> - reclaim with the knowledge of containers that are over limit
>   (Dave Hansen's patches + avoid overhead of combing the list)
> - have two lists one for the system and one per container
>

I will look at Dave's patch. Having two different list is not the right
approach. I will add some reclaim logic in kernel reclaim path.

Any idea why current RG implementation is not in mainline? Any effort
in reviving that and getting it in Andrew's tree.

> >
> > > > 6. Both active and inactive pages use physical pages. But, the
> > > >   controller only counts active pages and not inactive pages. why ?
> > > >
> > > The thought is, it is okay for containers to go over its limit as long
> > >
> > > Real number of "physical pages" used by the container is the sum of
> > > active and inactive pages.
> > >
> >

> > >From the user pov, the real sum of pages that are used by container for
> > user land is anon + file. Now some times it is possible that there are
> > active pages that are neither in page cache nor in use as anon.
> >
> >
> > > My question is, shouldn't that be used to check against page limit
> > > instead of active pages alone ?
> > I can use active+inactive as the test. Sure. But I will have to also
> > still have a check to make sure that number of active pages themselves
> > is not bigger than page_limit.
> >
> > > How do we describe "page limit" as (to the user) ?
> > >
> >
> > Amount of memory below which no container throttling will happen. And
>
> But, from the implementation one cannot clearly derive what we mean by
> "memory" here (physical ?, file + anon ?; if we say physical, it is not
> correct).
>

It is mostly correct. i.e. anon+file == total user physical memory for
container. Except for the corner cases when there could be stale
pagecache pages that are no longer on page cache but still on LRU (not
yet recalimed).

> > if the system is properly configured that it also ensures that this much
> > memory will always be there to user. If a container goes over this
> > limit then it will be throttled and it will suffer performance.
>
> But, the user's expectation would be that we would be throwing out pages
> based on LRU (within that container). But this implementation doesn't
> provide that behavior. It doesn't care about the working set.
>

IMO, user is expected to live inside the limits when containers are
defined. If the limits are exceeded then some performance impact will
happen. Having said that though I would still like to get some
optimizations in memory handler so that more appropriate pages are
deactivated.

> Performance impact will be lesser if we consider the working set and
> throw out pages based on LRU (within a container).
>
I don't deny it. But two separate LRUs is not an option.

> >
> > > > as there is enough memory in the system. When there is any memory

> > > > pressure then the inactive (+ dereferenced) pages get swapped out thus
> > > > penalizing the container. I'm also thinking of having hard limit for
> > >
> > > Reclamation goes through active pages and page cache pages before it
> > > gets into inactive pages. So, this may not work as you are explaining.
> >
> > That is a good point. I'll have to make a check in reclaim so that when
> > the system is ready for swap or write back then containers are looked
> > first.
> >
> > >
> > > > anonymous pages beyond which the container will not be able to grow its
> > > > anonymous pages.
> > >
> > > You might break the current behavior (memory pressure must be very high
> > > before these starts failing) if you are going to be strict about it.
> > >
> >
> > That feature when implemented will be a container specific.
>
> My point is, even though it is container specific, the behavior (inside
> a container) should be same as what a user sees at the system level now.
>
> For example, consider a workload that is run on a 1G system now, and
> user sees only occasional memory allocation failures and just a handful
> of oom kills. When the workload is moved to a container with 1G,
> failures the user see should be in the same order (and similar with
> performance characteristics).
>
> Do you agree that it will be the user's expectation ?
>

That will be nice to have feature. And for that any container
implementation will have to be as tightly intertwined with rest of vm as
cpuset is.

thanks,
-rohit
>

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Balbir Singh on Thu, 28 Sep 2006 21:53:44 GMT
View Forum Message <> Reply to Message

Rohit Seth wrote:
> On Thu, 2006-09-28 at 13:31 +0530, Balbir Singh wrote:
>> Chandra Seetharaman wrote:

>>> On Wed, 2006-09-27 at 14:28 -0700, Rohit Seth wrote:
>>>
>>> Rohit,
>>>
>>> For 1-4, I understand the rationale. But, your implementation deviates
>>> from the current behavior of the VM subsystem which could affect the
>>> ability of these patches getting into mainline.
>>>
>>> IMO, the current behavior in terms of reclamation, LRU, vm_swappiness,
>>> and writeback logic should be maintained.
>>>
>> <snip>
>>
>> Hi, Rohit,
>>
>> I have been playing around with the containers patch. I finally got
>> around to reading the code.
>>
>>
>> 1. Comments on reclaiming
>>
>> You could try the following options to overcome some of the disadvantages of the
>> current scheme.
>>
>> (a) You could consider a reclaim path based on Dave Hansen's Challenged memory
>> controller (see http://marc.theaimsgroup.com/?l=linux-mm&m=1155669825323 45&w=2).
>>
>
> I will go through that. Did you get a chance to stress the system and
> found any short comings that should be resolved.

I am yet to reach that stage, most of my playing around was w.r.t moving tasks.
At this point my basic aim was to test basic stability and look at the
accounting w.r.t correctness. I am yet to run any stress on the system.

>
>> (b) The other option is to do what the resource group memory controller does -
>> build a per group LRU list of pages (active, inactive) and reclaim
>> them using the existing code (by passing the correct container pointer,
>> instead of the zone pointer). One disadvantage of this approach is that
>> the global reclaim is impacted as the global LRU list is broken. At the
>> expense of another list, we could maintain two lists, global LRU and
>> container LRU lists. Depending on the context of the reclaim - (container
>> over limit, memory pressure) we could update/manipulate both lists.
>> This approach is definitely very expensive.
>>
>

> Two LRUs is a nice idea. Though I don't think it will go too far. It
> will involve adding another list pointers in the page structure. I
> agree that the mem handler is not optimal at all but I don't want to
> make it mimic kernel reclaimer at the same time.

One possible solution is to move the container tracking out of the pages and
into address_space and anon_vma. I guess this functionality will complicate
task migration and accounting a bit though.

>
>> 2. Comments on task migration support
>>
>> (a) One of the issues I found while using the container code is that, one could
>> add a task to a container say "a". "a" gets charged for the tasks usage,
>> when the same task moves to a different container say "b", when the task
>> exits, the credit goes to "b" and "a" remains indefinitely charged.
>>
> hmm, when the task is removed from "a" then "a" gets the credits for the
> amount of anon memory that is used by the task. Or do you mean
> something different.

Aah, I see. Once possible minor concern here is that a task could hope across
several containers, it could map files in each container and allocate page
cache pages, when it reaches the limit, it could hop to another container
and carry on until it hits the limit there.

>
>> (b) For tasks addition and removal, I think it's probably better to move
>> the entire process (thread group) rather than allow each individual thread
>> to move across containers. Having threads belonging to the same process
>> reside in different containers can be complex to handle, since they
>> share the same VM. Do you have a scenario where the above condition
>> would be useful?
>>
>>
> I don't have a scenario where a task actually gets to move out of
> container (except exit). That asynchronous removal of tasks has already
> got the code very complicated for locking etc. But if you think moving
> a thread group is useful then I will add that functionality.
>

Yes, that would be nice.

> Thanks,
> -rohit
>

--

Balbir Singh,
Linux Technology Center,
IBM Software Labs

Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Fri, 29 Sep 2006 00:22:59 GMT
View Forum Message <> Reply to Message

On Fri, 2006-09-29 at 03:23 +0530, Balbir Singh wrote:
> Rohit Seth wrote:
> > On Thu, 2006-09-28 at 13:31 +0530, Balbir Singh wrote:

> >
> >> (b) The other option is to do what the resource group memory controller does -
> >> build a per group LRU list of pages (active, inactive) and reclaim
> >> them using the existing code (by passing the correct container pointer,
> >> instead of the zone pointer). One disadvantage of this approach is that
> >> the global reclaim is impacted as the global LRU list is broken. At the
> >> expense of another list, we could maintain two lists, global LRU and
> >> container LRU lists. Depending on the context of the reclaim - (container
> >> over limit, memory pressure) we could update/manipulate both lists.
> >> This approach is definitely very expensive.
> >>
> >
> > Two LRUs is a nice idea. Though I don't think it will go too far. It
> > will involve adding another list pointers in the page structure. I
> > agree that the mem handler is not optimal at all but I don't want to
> > make it mimic kernel reclaimer at the same time.
>
> One possible solution is to move the container tracking out of the pages and
> into address_space and anon_vma. I guess this functionality will complicate
> task migration and accounting a bit though.
>

In the next version, I'm removing the per page pointer for container.
address_space already has a container pointer, I'm adding a pointer in
anon_vma as well. And that does seem to be complicating the accounting
just a wee bit. Though on its own, it is not helping the reclaim part.

I'll have to see how to handle kernel pages w/o a per page pointer.

> >
> >> 2. Comments on task migration support
> >>
> >> (a) One of the issues I found while using the container code is that, one could
> >> add a task to a container say "a". "a" gets charged for the tasks usage,

> >> when the same task moves to a different container say "b", when the task
> >> exits, the credit goes to "b" and "a" remains indefinitely charged.
> >>
> > hmm, when the task is removed from "a" then "a" gets the credits for the
> > amount of anon memory that is used by the task. Or do you mean
> > something different.
>
> Aah, I see. Once possible minor concern here is that a task could hope across
> several containers, it could map files in each container and allocate page
> cache pages, when it reaches the limit, it could hop to another container
> and carry on until it hits the limit there.
>
If there are multiple containers that a process can hop to then yes that
will happen.

-rohit
