

---

Subject: [patch02/05]: Containers(V2)- Generic Linux kernel changes

Posted by [Rohit Seth](#) on Wed, 20 Sep 2006 02:18:42 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

This patch contains changes to generic part of kernel code. These changes tracks events like new task creating, task's exit, new page allocation (both file and anonymous) etc.

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

```
fs/inode.c          | 3 +++
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/exit.c       | 2 ++
kernel/fork.c       | 9 ++++++++
mm/Makefile         | 2 ++
mm/filemap.c        | 4 +++++
mm/page_alloc.c     | 3 +++
mm/rmap.c           | 8 ++++++++
mm/swap.c           | 1 +
mm/vmscan.c         | 1 +
15 files changed, 60 insertions(+), 1 deletion(-)
```

```
--- linux-2.6.18-rc6-mm2.org/include/linux/mm_inline.h 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/include/linux/mm_inline.h 2006-09-14 15:09:08.000000000 -0700
@@ -1,9 +1,11 @@
+#include <linux/container.h>
```

```
static inline void
add_page_to_active_list(struct zone *zone, struct page *page)
{
    list_add(&page->lru, &zone->active_list);
    zone->nr_active++;
+ container_inc_activepage_count(page);
}
```

```
static inline void
@@ -23,6 +25,7 @@ add_page_to_inactive_list_tail(struct zo
static inline void
del_page_from_active_list(struct zone *zone, struct page *page)
{
+ container_dec_activepage_count(page);
    list_del(&page->lru);
```

```

zone->nr_active--;
}
@@ -41,6 +44,7 @@ del_page_from_lru(struct zone *zone, str
if (PageActive(page)) {
__ClearPageActive(page);
zone->nr_active--;
+ container_dec_activepage_count(page);
} else {
zone->nr_inactive--;
}
--- linux-2.6.18-rc6-mm2.org/include/linux/fs.h 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/include/linux/fs.h 2006-09-14 15:09:08.000000000 -0700
@@ -449,6 +449,7 @@ struct address_space_operations {
};

struct backing_dev_info;
+struct container_struct;
struct address_space {
struct inode *host; /* owner: inode, block_device */
struct radix_tree_root page_tree; /* radix tree of all pages */
@@ -465,6 +466,10 @@ struct address_space {
struct backing_dev_info *backing_dev_info; /* device readahead, etc */
spinlock_t private_lock; /* for use by the address_space */
struct list_head private_list; /* ditto */
+#ifdef CONFIG_CONTAINERS
+ struct container_struct *ctn; /* Pointer to container */
+ struct list_head ctn_mapping_list; /* List of files belonging to same container*/
+#endif
struct address_space *assoc_mapping; /* ditto */
} __attribute__((aligned(sizeof(long))));
/*
--- linux-2.6.18-rc6-mm2.org/include/linux/sched.h 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/include/linux/sched.h 2006-09-14 15:09:08.000000000 -0700
@@ -298,6 +298,8 @@ typedef unsigned long mm_counter_t;
(mm)->hiwater_vm = (mm)->total_vm; \
} while (0)

+struct container_struct;
+
struct mm_struct {
struct vm_area_struct * mmap; /* list of VMAs */
struct rb_root mm_rb;
@@ -1046,6 +1048,10 @@ struct task_struct {
#ifdef CONFIG_TASK_DELAY_ACCT
struct task_delay_info *delays;
#endif
+#ifdef CONFIG_CONTAINERS
+ struct container_struct *ctn;

```

```

+ struct list_head ctn_task_list; /*List of processes belonging to container */
+#endif
};

static inline pid_t process_group(struct task_struct *tsk)
--- linux-2.6.18-rc6-mm2.org/include/linux/mm_types.h 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/include/linux/mm_types.h 2006-09-14 15:15:30.000000000 -0700
@@ -6,6 +6,7 @@
#include <linux/list.h>
#include <linux/spinlock.h>

+struct container_struct;
struct address_space;

/*
@@ -48,6 +49,9 @@ struct page {
    struct list_head lru; /* Pageout list, eg. active_list
        * protected by zone->lru_lock !
        */
+#ifdef CONFIG_CONTAINERS
+ struct container_struct *ctn; /* Pointer to container, may be NULL */
+#endif
/*
    * On machines where all RAM is mapped into kernel address space,
    * we can simply calculate the virtual address. On machines with
--- linux-2.6.18-rc6-mm2.org/mm/filemap.c 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/mm/filemap.c 2006-09-14 15:09:08.000000000 -0700
@@ -30,6 +30,7 @@
#include <linux/security.h>
#include <linux/syscalls.h>
#include <linux/cpuset.h>
+#include <linux/container.h>
#include "filemap.h"
#include "internal.h"

@@ -126,6 +127,7 @@ void __remove_from_page_cache(struct pag
    page->mapping = NULL;
    mapping->npages--;
    __dec_zone_page_state(page, NR_FILE_PAGES);
+ container_dec_filepage_count(page);
}
EXPORT_SYMBOL(__remove_from_page_cache);

@@ -461,6 +463,8 @@ int add_to_page_cache(struct page *page,
}
write_unlock_irq(&mapping->tree_lock);
radix_tree_preload_end();
+ if (!error)

```

```

+ container_inc_filepage_count(mapping, page);
}
return error;
}
--- linux-2.6.18-rc6-mm2.org/init/Kconfig 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/init/Kconfig 2006-09-18 10:48:08.000000000 -0700
@@ -560,6 +560,14 @@ config STOP_MACHINE
depends on (SMP && MODULE_UNLOAD) || HOTPLUG_CPU
help
Need stop_machine() primitive.
+
+config CONTAINERS
+ bool "Containers"
+ def_bool y
+ depends on CONFIGFS_FS
+ help
+ This option allows grouping of resources like memory and tasks. It
+ depends on CONFIGFS_FS support in psuedo filesystem.
endmenu

menu "Block layer"
--- linux-2.6.18-rc6-mm2.org/mm/Makefile 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/mm/Makefile 2006-09-14 15:09:08.000000000 -0700
@@ -29,3 +29,5 @@ obj-$(CONFIG_MEMORY_HOTPLUG) += memory_h
obj-$(CONFIG_FS_XIP) += filemap_xip.o
obj-$(CONFIG_MIGRATION) += migrate.o
obj-$(CONFIG_SMP) += allocpercpu.o
+obj-$(CONFIG_CONTAINERS) += container.o container_mm.o
+
--- linux-2.6.18-rc6-mm2.org/mm/page_alloc.c 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/mm/page_alloc.c 2006-09-14 15:09:08.000000000 -0700
@@ -39,6 +39,7 @@
#include <linux/stop_machine.h>
#include <linux/sort.h>
#include <linux/pfn.h>
+#include <linux/container.h>

#include <asm/tlbflush.h>
#include <asm/div64.h>
@@ -502,6 +503,7 @@ static void free_one_page(struct zone *z
zone->pages_scanned = 0;
__free_one_page(page, zone, order);
spin_unlock(&zone->lock);
+ container_init_page_ptr(page, NULL);
}

static void __free_pages_ok(struct page *page, unsigned int order)
@@ -798,6 +800,7 @@ static void fastcall free_hot_cold_page(

```

```

arch_free_page(page, 0);

+ container_init_page_ptr(page, NULL);
  if (PageAnon(page))
    page->mapping = NULL;
  if (free_pages_check(page))
--- linux-2.6.18-rc6-mm2.org/mm/rmap.c 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/mm/rmap.c 2006-09-14 15:09:08.000000000 -0700
@@ -53,6 +53,7 @@
#include <linux/rmap.h>
#include <linux/rcupdate.h>
#include <linux/module.h>
+#include <linux/container.h>

#include <asm/tlbflush.h>

@@ -521,6 +522,7 @@ static void __page_set_anon_rmap(struct
 * interrupts because it is not modified via interrupt.
 */
__inc_zone_page_state(page, NR_ANON_PAGES);
+ container_inc_page_count(page);
}

/**
@@ -563,8 +565,10 @@ void page_add_new_anon_rmap(struct page
 */
void page_add_file_rmap(struct page *page)
{
- if (atomic_inc_and_test(&page->_mapcount))
+ if (atomic_inc_and_test(&page->_mapcount)) {
  __inc_zone_page_state(page, NR_FILE_MAPPED);
+ container_inc_page_count(page);
+ }
}

/**
@@ -598,6 +602,8 @@ void page_remove_rmap(struct page *page)
  set_page_dirty(page);
  __dec_zone_page_state(page,
  PageAnon(page) ? NR_ANON_PAGES : NR_FILE_MAPPED);
+ container_dec_page_count(page);
+
}
}

--- linux-2.6.18-rc6-mm2.org/mm/swap.c 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/mm/swap.c 2006-09-14 15:09:08.000000000 -0700

```

```

@@ -196,6 +196,7 @@ void fastcall lru_cache_add_active(struct
    struct pagevec *pvec = &get_cpu_var(lru_add_active_pvecs);

    page_cache_get(page);
+ container_init_page_ptr(page, current);
    if (!pagevec_add(pvec, page))
        __pagevec_lru_add_active(pvec);
    put_cpu_var(lru_add_active_pvecs);
--- linux-2.6.18-rc6-mm2.org/mm/vmscan.c 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/mm/vmscan.c 2006-09-14 15:09:08.000000000 -0700
@@ -818,6 +818,7 @@ force_reclaim_mapped:
    SetPageLRU(page);
    VM_BUG_ON(!PageActive(page));
    ClearPageActive(page);
+ container_dec_activepage_count(page);

    list_move(&page->lru, &zone->inactive_list);
    pgmoved++;
--- linux-2.6.18-rc6-mm2.org/kernel/exit.c 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/kernel/exit.c 2006-09-18 18:18:22.000000000 -0700
@@ -41,6 +41,7 @@
#include <linux/audit.h> /* for audit_free() */
#include <linux/resource.h>
#include <linux/blkdev.h>
+#include <linux/container.h>

#include <asm/uaccess.h>
#include <asm/unistd.h>
@@ -171,6 +172,7 @@ repeat:

    sched_exit(p);
    write_unlock_irq(&tasklist_lock);
+ container_remove_task(p, NULL);
    proc_flush_task(p);
    release_thread(p);
    call_rcu(&p->rcu, delayed_put_task_struct);
--- linux-2.6.18-rc6-mm2.org/kernel/fork.c 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/kernel/fork.c 2006-09-19 10:36:20.000000000 -0700
@@ -47,6 +47,7 @@
#include <linux/cn_proc.h>
#include <linux/delayacct.h>
#include <linux/taskstats_kern.h>
+#include <linux/container.h>
#include <linux/random.h>

#include <asm/pgtable.h>
@@ -175,6 +176,13 @@ static struct task_struct *dup_task_stru
}

```

```

    *tsk = *orig;
+
+ container_init_task_ptr(tsk);
+ if (container_add_task(tsk, orig, NULL) == -ENOSPC) {
+ free_task_struct(tsk);
+ free_thread_info(ti);
+ return NULL;
+ }
    tsk->thread_info = ti;
    setup_thread_stack(tsk, orig);

@@ -1295,6 +1303,7 @@ bad_fork_cleanup_count:
    atomic_dec(&p->user->processes);
    free_uid(p->user);
bad_fork_free:
+ container_remove_task(p, NULL);
    free_task(p);
fork_out:
    return ERR_PTR(retval);
--- linux-2.6.18-rc6-mm2.org/kernel/Makefile 2006-09-14 15:28:33.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/kernel/Makefile 2006-09-14 15:09:08.000000000 -0700
@@ -52,6 +52,7 @@ obj-$(CONFIG_RELAY) += relay.o
obj-$(CONFIG_UTS_NS) += utsname.o
obj-$(CONFIG_TASK_DELAY_ACCT) += delayacct.o
obj-$(CONFIG_TASKSTATS) += taskstats.o tsacct.o
+obj-$(CONFIG_CONTAINERS) += container_configs.o

ifneq ($(CONFIG_SCHED_NO_NO_OMIT_FRAME_POINTER),y)
# According to Alan Modra <alan@linuxcare.com.au>, the -fno-omit-frame-pointer is
--- linux-2.6.18-rc6-mm2.org/fs/inode.c 2006-09-14 15:28:31.000000000 -0700
+++ linux-2.6.18-rc6-mm2.ctn/fs/inode.c 2006-09-14 15:09:08.000000000 -0700
@@ -22,6 +22,7 @@
#include <linux/bootmem.h>
#include <linux/inotify.h>
#include <linux/mount.h>
+#include <linux/container.h>

/*
 * This is needed for the following functions:
@@ -164,6 +165,7 @@ static struct inode *alloc_inode(struct
}
inode->i_private = 0;
inode->i_mapping = mapping;
+ container_add_file(mapping, NULL);
}
return inode;
}

```

```
@@ -172,6 +174,7 @@ void destroy_inode(struct inode *inode)
{
    BUG_ON(inode_has_buffers(inode));
    security_inode_free(inode);
+ container_remove_file(inode->i_mapping);
    if (inode->i_sb->s_op->destroy_inode)
        inode->i_sb->s_op->destroy_inode(inode);
    else
```

---

---

Subject: Re: [patch02/05]: Containers(V2)- Generic Linux kernel changes  
Posted by [Andi Kleen](#) on Wed, 20 Sep 2006 11:27:33 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Rohit Seth <rohitseth@google.com> writes:

```
> */
> +#ifdef CONFIG_CONTAINERS
> + struct container_struct *ctn; /* Pointer to container, may be NULL */
> +#endif
```

I still don't think it's a good idea to add a pointer to struct page for this.  
This means any kernel that enables the config would need to carry this significant overhead, no matter if containers are used to not.

Better would be to store them in some other data structure that is only allocated on demand or figure out a way to store them in the sometimes not all used fields in struct page.

BTW your patchkit seems to be also in wrong order in that when 02 is applied it won't compile.

-Andi

---

---

Subject: Re: [patch02/05]: Containers(V2)- Generic Linux kernel changes  
Posted by [Rohit Seth](#) on Wed, 20 Sep 2006 16:44:29 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 2006-09-20 at 13:27 +0200, Andi Kleen wrote:

```
> Rohit Seth <rohitseth@google.com> writes:
> > */
> > +#ifdef CONFIG_CONTAINERS
> > + struct container_struct *ctn; /* Pointer to container, may be NULL */
> > +#endif
>
> I still don't think it's a good idea to add a pointer to struct page for this.
```



I thought last time you supported adding a pointer to struct page (when you mentioned next gen slab will also consume page->mapping). May be I missed your point.

> This means any kernel that enables the config would need to carry this significant overhead, no matter if containers are used to not.

>  
> Sure this is non-zero overhead but I think this is the logical place to track the memory.

> Better would be to store them in some other data structure that is only allocated on demand or figure out a way to store them in the sometimes not all used fields in struct page.

>

which one...I think the fields in page structure are already getting doubly used.

> BTW your patchkit seems to be also in wrong order in that when 02 is applied it won't compile.

Not sure if I understood that. I've myself tested these patches on 2.6.18-rc6-mm2 kernel and they apply just fine. Are you just trying to apply 02....if so then that wouldn't suffice.

-rohit

---

Subject: Re: [patch02/05]: Containers(V2)- Generic Linux kernel changes  
Posted by [Andi Kleen](#) on Wed, 20 Sep 2006 18:14:48 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Wednesday 20 September 2006 18:44, Rohit Seth wrote:

> On Wed, 2006-09-20 at 13:27 +0200, Andi Kleen wrote:

> > Rohit Seth <rohitseth@google.com> writes:

> > > \*/

> > > +**#ifdef CONFIG\_CONTAINERS**

> > > + struct container\_struct \*ctn; /\* Pointer to container, may be NULL \*/

> > > +**#endif**

> >

> > I still don't think it's a good idea to add a pointer to struct page for this.

>

> I thought last time you supported adding a pointer to struct page (when you mentioned next gen slab will also consume page->mapping).

I didn't. Alternative was a separate data structure.

> which one...I think the fields in page structure are already getting

> doubly used.

There are lots of different cases. At least for anonymous memory  
->mapping should be free. Perhaps that could be used for anonymous  
memory and a separate data structure for the important others.

slab should have at least one field free too, although it might be a different  
one (iirc Christoph's rewrite uses more than the current slab, but it would  
surprise me if he needed all)

> > BTW your patchkit seems to be also in wrong order in that when 02 is applied  
> > it won't compile.

>

> Not sure if I understood that. I've myself tested these patches on  
> 2.6.18-rc6-mm2 kernel and they apply just fine. Are you just trying to  
> apply 02....if so then that wouldn't suffice.

I meant assuming the patchkit was applied you would break binary search  
inbetween because not each piece compiles on its own.

-Andi

---

Subject: Re: [patch02/05]: Containers(V2)- Generic Linux kernel changes  
Posted by [Rohit Seth](#) on Wed, 20 Sep 2006 18:19:37 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 2006-09-20 at 20:14 +0200, Andi Kleen wrote:

> On Wednesday 20 September 2006 18:44, Rohit Seth wrote:

> > On Wed, 2006-09-20 at 13:27 +0200, Andi Kleen wrote:

> > > Rohit Seth <rohitseth@google.com> writes:

> > > > \*/

> > > > +#ifdef CONFIG\_CONTAINERS

> > > > + struct container\_struct \*ctn; /\* Pointer to container, may be NULL \*/

> > > > +#endif

> > >

> > > I still don't think it's a good idea to add a pointer to struct page for this.

> >

> > I thought last time you supported adding a pointer to struct page (when  
> > you mentioned next gen slab will also consume page->mapping).

>

> I didn't. Alternative was a separate data structure.

>

> > which one...I think the fields in page structure are already getting

> > doubly used.

>

> There are lots of different cases. At least for anonymous memory

> ->mapping should be free. Perhaps that could be used for anonymous

> memory and a separate data structure for the important others.  
>

It is not free for anonymous memory as it is overloaded with pointer to anon\_vma. I think one single pointer consistent across all page usages is just so much cleaner and simple...

> slab should have at least one field free too, although it might be a different  
> one (iirc Christoph's rewrite uses more than the current slab, but it would  
> surprise me if he needed all)

>  
>>> BTW your patchkit seems to be also in wrong order in that when 02 is applied  
>>> it won't compile.

>>  
>> Not sure if I understood that. I've myself tested these patches on  
>> 2.6.18-rc6-mm2 kernel and they apply just fine. Are you just trying to  
>> apply 02....if so then that wouldn't suffice.

>  
> I meant assuming the patchkit was applied you would break binary search  
> inbetween because not each piece compiles on its own.

I've currently separated the patches based on where the changes are made and something that makes each a logical block. I will reorder the patches next time so that each subsequent patch compiles.

-rohit

---

Subject: Re: [patch02/05]: Containers(V2)- Generic Linux kernel changes  
Posted by [Andi Kleen](#) on Wed, 20 Sep 2006 18:32:59 GMT  
[View Forum Message](#) <> [Reply to Message](#)

> It is not free for anonymous memory as it is overloaded with pointer to  
> anon\_vma.

Sorry. But ->index should be free there.

> I think one single pointer consistent across all page usages  
> is just so much cleaner and simple...

It just costs you 0.2% of all your memory in all cases. Too much imho.

-Andi

---

Subject: Re: [patch02/05]: Containers(V2)- Generic Linux kernel changes

Posted by [Christoph Lameter](#) on Thu, 21 Sep 2006 00:23:50 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 20 Sep 2006, Andi Kleen wrote:

> There are lots of different cases. At least for anonymous memory  
> ->mapping should be free. Perhaps that could be used for anonymous  
> memory and a separate data structure for the important others.

mapping is used for swap and to point to the anon vma.

> slab should have at least one field free too, although it might be a different  
> one (iirc Christoph's rewrite uses more than the current slab, but it would  
> surprise me if he needed all)

slab currently has lots of fields free but my rewrite uses all of them.  
And AFAICT this patchset does not track slab pages.

Hmm.... Build a radix tree with pointers to the pages?

---

---

Subject: Re: [patch02/05]: Containers(V2)- Generic Linux kernel changes

Posted by [Rohit Seth](#) on Thu, 21 Sep 2006 00:37:53 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 2006-09-20 at 17:23 -0700, Christoph Lameter wrote:

> On Wed, 20 Sep 2006, Andi Kleen wrote:

>  
>> There are lots of different cases. At least for anonymous memory  
>> ->mapping should be free. Perhaps that could be used for anonymous  
>> memory and a separate data structure for the important others.

>  
> mapping is used for swap and to point to the anon vma.

>  
>> slab should have at least one field free too, although it might be a different  
>> one (iirc Christoph's rewrite uses more than the current slab, but it would  
>> surprise me if he needed all)

>  
> slab currently has lots of fields free but my rewrite uses all of them.  
> And AFAICT this patchset does not track slab pages.

>

Currently it doesn't track kernel memory. That is why I don't want to  
over load any of the existing fields.

> Hmm.... Build a radix tree with pointers to the pages?

>

...my preference is to leave it in page struct...that has non-zero cost.

-rohit

---

---

Subject: Re: [patch02/05]: Containers(V2)- Generic Linux kernel changes  
Posted by [Christoph Lameter](#) on Thu, 21 Sep 2006 00:41:23 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 20 Sep 2006, Rohit Seth wrote:

> ...my preference is to leave it in page struct...that has non-zero cost.

Oh. Making struct page bigger than a cacheline or not fitting exactly in a cacheline has some costs.

---

---

Subject: Re: [patch02/05]: Containers(V2)- Generic Linux kernel changes  
Posted by [Rohit Seth](#) on Thu, 21 Sep 2006 00:53:40 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 2006-09-20 at 17:41 -0700, Christoph Lameter wrote:

> On Wed, 20 Sep 2006, Rohit Seth wrote:

>

> > ...my preference is to leave it in page struct...that has non-zero cost.

>

> Oh. Making struct page bigger than a cacheline or not fitting exactly in a

> cacheline has some costs.

>

Isn't most of the architectures not defining WANT\_PAGE\_VIRTUAL and thus the page structure on these is not fitting cache line well. In most of these cases the additional pointer will actually make the structure 64 bytes on 64-bit machines.

-rohit

---

---

Subject: Re: [ckrm-tech] [patch02/05]: Containers(V2)- Generic Linux kernel changes

Posted by [Chandra Seetharaman](#) on Thu, 21 Sep 2006 01:33:32 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 2006-09-20 at 17:23 -0700, Christoph Lameter wrote:

> On Wed, 20 Sep 2006, Andi Kleen wrote:

>

> > There are lots of different cases. At least for anonymous memory

---

> > ->mapping should be free. Perhaps that could be used for anonymous  
> > memory and a separate data structure for the important others.  
>  
> mapping is used for swap and to point to the anon vma.  
>  
> > slab should have at least one field free too, although it might be a different  
> > one (iirc Christoph's rewrite uses more than the current slab, but it would  
> > surprise me if he needed all)  
>  
> slab currently has lots of fields free but my rewrite uses all of them.  
> And AFAICT this patchset does not track slab pages.  
>  
> Hmm.... Build a radix tree with pointers to the pages?

Yes, that would be a way to isolate the overhead.

--

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

---

Subject: Re: [ckrm-tech] [patch02/05]: Containers(V2)- Generic Linux kernel changes

Posted by [Rohit Seth](#) on Thu, 21 Sep 2006 22:29:45 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 2006-09-20 at 18:33 -0700, Chandra Seetharaman wrote:

> On Wed, 2006-09-20 at 17:23 -0700, Christoph Lameter wrote:

> > On Wed, 20 Sep 2006, Andi Kleen wrote:

> >

> > > There are lots of different cases. At least for anonymous memory

> > > ->mapping should be free. Perhaps that could be used for anonymous

> > > memory and a separate data structure for the important others.

> >

> > mapping is used for swap and to point to the anon vma.

> >

> > > slab should have at least one field free too, although it might be a different

> > > one (iirc Christoph's rewrite uses more than the current slab, but it would

> > > surprise me if he needed all)

> >

> > slab currently has lots of fields free but my rewrite uses all of them.

> > And AFAICT this patchset does not track slab pages.

> >

> > Hmm.... Build a radix tree with pointers to the pages?

>

> Yes, that would be a way to isolate the overhead.

>

As said earlier, the additional cost is not really that much. And if we do get additional benefit of proper cache alignment for page struct because of this additional pointer then it could be a almost zero cost.

And in mm kernel, I also see PAGE\_OWNER defined which puts the whole page struct out of sync with caches.

Christoph, Please let me know if there is still a placeholder for container pointer ONLY for kernel pages. I think it is easy to do away with per page pointer for user land as it can be stored in anon\_vma or mappings.

-rohit

---