
Subject: [PATCH 2.6.24-rc3-mm1] IPC: make struct ipc_ids static in ipc_namespace
Posted by [Pierre Peiffer](#) on Thu, 22 Nov 2007 14:54:48 GMT
[View Forum Message](#) <> [Reply to Message](#)

Each ipc_namespace contains a table of 3 pointers to struct ipc_ids (3 for msg, sem and shm, structure used to store all ipcs)
These 'struct ipc_ids' are dynamically allocated for each ipc_namespace as the ipc_namespace itself (for the init namespace, they are initialized with pointers to static variables instead)

It is so for historical reason: in fact, before the use of idr to store the ipcs, the ipcs were stored in tables of variable length, depending of the maximum number of ipc allowed.
Now, these 'struct ipc_ids' have a fixed size. As they are allocated in any cases for each new ipc_namespace, there is no gain of memory in having them allocated separately of the struct ipc_namespace.

This patch proposes to make this table static in the struct ipc_namespace.
Thus, we can allocate all in once and get rid of all the code needed to allocate and free these ipc_ids separately.

Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>

include/linux/ipc_namespace.h | 13 ++++++++
ipc/msg.c | 26 +++++-----
ipc/namespace.c | 25 +++++-----
ipc/sem.c | 26 +++++-----
ipc/shm.c | 26 +++++-----
ipc/util.c | 6 +++--
ipc/util.h | 16 +++++-----
7 files changed, 34 insertions(+), 104 deletions(-)

Index: b/include/linux/ipc_namespace.h

```
=====
--- a/include/linux/ipc_namespace.h
+++ b/include/linux/ipc_namespace.h
@@ -2,11 +2,20 @@
#define __IPC_NAMESPACE_H__

#include <linux/err.h>
#include <linux/idr.h>
#include <linux/rwsem.h>
+
+struct ipc_ids {
+ int in_use;
+ unsigned short seq;
+ unsigned short seq_max;
+ struct rw_semaphore rw_mutex;
```

```

+ struct idr ipcs_idr;
+};

-struct ipc_ids;
struct ipc_namespace {
    struct kref kref;
- struct ipc_ids *ids[3];
+ struct ipc_ids ids[3];

    int sem_ctls[4];
    int used_sems;
Index: b/ipc/msg.c
=====
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -67,9 +67,7 @@ struct msg_sender {
#define SEARCH_NOTEQUAL 3
#define SEARCH_LESSEQUAL 4

-static struct ipc_ids init_msg_ids;
-
-#define msg_ids(ns) ((ns)->ids[IPC_MSG_IDS])
+#define msg_ids(ns) ((ns)->ids[IPC_MSG_IDS])

#define msg_unlock(msq) ipc_unlock(&(msq)->q_perm)
#define msg_buildid(id, seq) ipc_buildid(id, seq)
@@ -80,30 +78,17 @@ static int newque(struct ipc_namespace *
static int sysvipc_msg_proc_show(struct seq_file *s, void *it);
#endif

-static void __msg_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)
+void msg_init_ns(struct ipc_namespace *ns)
{
- ns->ids[IPC_MSG_IDS] = ids;
  ns->msg_ctlmax = MSGMAX;
  ns->msg_ctlmnb = MSGMNB;
  ns->msg_ctlmni = MSGMNI;
  atomic_set(&ns->msg_bytes, 0);
  atomic_set(&ns->msg_hdrs, 0);
- ipc_init_ids(ids);
+ ipc_init_ids(&ns->ids[IPC_MSG_IDS]);
}

#ifdef CONFIG_IPC_NS
-int msg_init_ns(struct ipc_namespace *ns)
-{
- struct ipc_ids *ids;
-

```

```

- ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
- if (ids == NULL)
- return -ENOMEM;
-
- __msg_init_ns(ns, ids);
- return 0;
-}
-
void msg_exit_ns(struct ipc_namespace *ns)
{
    struct msg_queue *msq;
@@ -126,15 +111,12 @@ void msg_exit_ns(struct ipc_namespace *n
}

    up_write(&msg_ids(ns).rw_mutex);
-
- kfree(ns->ids[IPC_MSG_IDS]);
- ns->ids[IPC_MSG_IDS] = NULL;
}
#endif

void __init msg_init(void)
{
- __msg_init_ns(&init_ipc_ns, &init_msg_ids);
+ msg_init_ns(&init_ipc_ns);
    ipc_init_proc_interface("sysvipc/msg",
        "      key      msqid perms      cbytes      qnum lspid lrpid  uid   gid   cuid   cgid    stime    rtime\n",
        ctime\n",
        IPC_MSG_IDS, sysvipc_msg_proc_show);
Index: b/ipc/namespace.c
=====
--- a/ipc/namespace.c
+++ b/ipc/namespace.c
@@ -14,35 +14,18 @@

static struct ipc_namespace *clone_ipc_ns(struct ipc_namespace *old_ns)
{
- int err;
    struct ipc_namespace *ns;

- err = -ENOMEM;
    ns = kmalloc(sizeof(struct ipc_namespace), GFP_KERNEL);
    if (ns == NULL)
- goto err_mem;
+ return ERR_PTR(-ENOMEM);

- err = sem_init_ns(ns);
- if (err)

```

```

- goto err_sem;
- err = msg_init_ns(ns);
- if (err)
- goto err_msg;
- err = shm_init_ns(ns);
- if (err)
- goto err_shm;
+ sem_init_ns(ns);
+ msg_init_ns(ns);
+ shm_init_ns(ns);

```

```

    kref_init(&ns->kref);
    return ns;

```

```

-
-err_shm:
- msg_exit_ns(ns);
-err_msg:
- sem_exit_ns(ns);
-err_sem:
- kfree(ns);
-err_mem:
- return ERR_PTR(err);
}

```

```

struct ipc_namespace *copy_ipcs(unsigned long flags, struct ipc_namespace *ns)
Index: b/ipc/sem.c

```

```

=====
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -87,14 +87,12 @@
#include <asm/uaccess.h>
#include "util.h"

#define sem_ids(ns) ((ns)->ids[IPC_SEM_IDS])
#define sem_ids(ns) ((ns)->ids[IPC_SEM_IDS])

#define sem_unlock(sma) ipc_unlock(&(sma)->sem_perm)
#define sem_checkid(sma, semid) ipc_checkid(&sma->sem_perm, semid)
#define sem_buildid(id, seq) ipc_buildid(id, seq)

-static struct ipc_ids init_sem_ids;
-
static int newary(struct ipc_namespace *, struct ipc_params *);
static void freeary(struct ipc_namespace *, struct sem_array *);
#ifdef CONFIG_PROC_FS
@@ -118,30 +116,17 @@ static int sysvipc_sem_proc_show(struct
#define sc_semopm sem_ctls[2]
#define sc_semmni sem_ctls[3]

```

```

-static void __sem_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)
+void sem_init_ns(struct ipc_namespace *ns)
{
- ns->ids[IPC_SEM_IDS] = ids;
  ns->sc_semmsl = SEMMSL;
  ns->sc_semmns = SEMMNS;
  ns->sc_semopm = SEMOPM;
  ns->sc_semmni = SEMMNI;
  ns->used_sems = 0;
- ipc_init_ids(ids);
+ ipc_init_ids(&ns->ids[IPC_SEM_IDS]);
}

#ifdef CONFIG_IPC_NS
-int sem_init_ns(struct ipc_namespace *ns)
-{
- struct ipc_ids *ids;
-
- ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
- if (ids == NULL)
- return -ENOMEM;
-
- __sem_init_ns(ns, ids);
- return 0;
-}
-
void sem_exit_ns(struct ipc_namespace *ns)
{
  struct sem_array *sma;
@@ -163,15 +148,12 @@ void sem_exit_ns(struct ipc_namespace *n
  total++;
}
up_write(&sem_ids(ns).rw_mutex);
-
- kfree(ns->ids[IPC_SEM_IDS]);
- ns->ids[IPC_SEM_IDS] = NULL;
}
#endif

void __init sem_init(void)
{
- __sem_init_ns(&init_ipc_ns, &init_sem_ids);
+ sem_init_ns(&init_ipc_ns);
  ipc_init_proc_interface("sysvipc/sem",
    "      key      semid perms   nsems uid  gid  cuid  cgid   otime   ctime\n",
    IPC_SEM_IDS, sysvipc_sem_proc_show);
Index: b/ipc/util.c

```

```

=====
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -782,7 +782,7 @@ static void *sysvipc_proc_next(struct se
    if (ipc && ipc != SEQ_START_TOKEN)
        ipc_unlock(ipc);

- return sysvipc_find_ipc(iter->ns->ids[iface->ids], *pos, pos);
+ return sysvipc_find_ipc(&iter->ns->ids[iface->ids], *pos, pos);
}

/*
@@ -795,7 +795,7 @@ static void *sysvipc_proc_start(struct s
    struct ipc_proc_iface *iface = iter->iface;
    struct ipc_ids *ids;

- ids = iter->ns->ids[iface->ids];
+ ids = &iter->ns->ids[iface->ids];

/*
    * Take the lock - this will be released by the corresponding
@@ -826,7 +826,7 @@ static void sysvipc_proc_stop(struct seq
    if (ipc && ipc != SEQ_START_TOKEN)
        ipc_unlock(ipc);

- ids = iter->ns->ids[iface->ids];
+ ids = &iter->ns->ids[iface->ids];
    /* Release the lock we took in start() */
    up_read(&ids->rw_mutex);
}
Index: b/ipc/util.h
=====
--- a/ipc/util.h
+++ b/ipc/util.h
@@ -10,7 +10,6 @@
#ifndef _IPC_UTIL_H
#define _IPC_UTIL_H

#include <linux/idr.h>
#include <linux/err.h>

#define USHRT_MAX 0xffff
@@ -22,22 +21,14 @@ void shm_init (void);

struct ipc_namespace;

-int sem_init_ns(struct ipc_namespace *ns);
-int msg_init_ns(struct ipc_namespace *ns);

```

```

-int shm_init_ns(struct ipc_namespace *ns);
+void sem_init_ns(struct ipc_namespace *ns);
+void msg_init_ns(struct ipc_namespace *ns);
+void shm_init_ns(struct ipc_namespace *ns);

void sem_exit_ns(struct ipc_namespace *ns);
void msg_exit_ns(struct ipc_namespace *ns);
void shm_exit_ns(struct ipc_namespace *ns);

-struct ipc_ids {
- int in_use;
- unsigned short seq;
- unsigned short seq_max;
- struct rw_semaphore rw_mutex;
- struct idr ipcs_idr;
-};
-
/*
 * Structure that holds the parameters needed by the ipc operations
 * (see after)
@@ -68,6 +59,7 @@ struct ipc_ops {
};

struct seq_file;
+struct ipc_ids;

void ipc_init_ids(struct ipc_ids *);
#ifdef CONFIG_PROC_FS
Index: b/ipc/shm.c
=====
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -56,9 +56,7 @@ struct shm_file_data {
static const struct file_operations shm_file_operations;
static struct vm_operations_struct shm_vm_ops;

-static struct ipc_ids init_shm_ids;
-
-#define shm_ids(ns) ((ns)->ids[IPC_SHM_IDS])
+#define shm_ids(ns) ((ns)->ids[IPC_SHM_IDS])

#define shm_unlock(shp) \
ipc_unlock(&(shp)->shm_perm)
@@ -72,14 +70,13 @@ static void shm_destroy (struct ipc_name
static int sysvipc_shm_proc_show(struct seq_file *s, void *it);
#endif

-static void __shm_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)

```

```

+void shm_init_ns(struct ipc_namespace *ns)
{
- ns->ids[IPC_SHM_IDS] = ids;
  ns->shm_ctlmax = SHMMAX;
  ns->shm_ctlall = SHMALL;
  ns->shm_ctlmni = SHMMNI;
  ns->shm_tot = 0;
- ipc_init_ids(ids);
+ ipc_init_ids(&ns->ids[IPC_SHM_IDS]);
}

/*
@@ -98,18 +95,6 @@ static void do_shm_rmid(struct ipc_names
}

#ifdef CONFIG_IPC_NS
-int shm_init_ns(struct ipc_namespace *ns)
-{
- struct ipc_ids *ids;
-
- ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
- if (ids == NULL)
- return -ENOMEM;
-
- __shm_init_ns(ns, ids);
- return 0;
-}
-
void shm_exit_ns(struct ipc_namespace *ns)
{
  struct shmid_kernel *shp;
@@ -131,15 +116,12 @@ void shm_exit_ns(struct ipc_namespace *n
  total++;
}
up_write(&shm_ids(ns).rw_mutex);
-
- kfree(ns->ids[IPC_SHM_IDS]);
- ns->ids[IPC_SHM_IDS] = NULL;
}
#endif

void __init shm_init (void)
{
- __shm_init_ns(&init_ipc_ns, &init_shm_ids);
+ shm_init_ns(&init_ipc_ns);
  ipc_init_proc_interface("sysvipc/shm",
    "      key      shmid perms      size cpid lpid nattch  uid  gid cuid cgid   atime   dtime\n",
    ctime\n",

```

IPC_SHM_IDS, sysvipc_shm_proc_show);

--

Pierre Peiffer

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: make struct ipc_ids static in ipc_namespace

Posted by [Cedric Le Goater](#) on Fri, 23 Nov 2007 07:26:45 GMT

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

Pierre Peiffer wrote:

>

> Each ipc_namespace contains a table of 3 pointers to struct ipc_ids (3 for
> msg, sem and shm, structure used to store all ipc's)
> These 'struct ipc_ids' are dynamically allocated for each ipc_namespace as
> the ipc_namespace itself (for the init namespace, they are initialized with
> pointers to static variables instead)

>

> It is so for historical reason: in fact, before the use of idr to store the
> ipc's, the ipc's were stored in tables of variable length, depending of the
> maximum number of ipc allowed.
> Now, these 'struct ipc_ids' have a fixed size. As they are allocated in any
> cases for each new ipc_namespace, there is no gain of memory in having them
> allocated separately of the struct ipc_namespace.

>

> This patch proposes to make this table static in the struct ipc_namespace.
> Thus, we can allocate all in once and get rid of all the code needed to
> allocate and free these ipc_ids separately.

It looks safe and saves quite a lot of line.

Pavel, what do you think of it ?

Acked-by: Cedric Le Goater <clg@fr.ibm.com>

Thanks,

C.

> Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>

> ---

> include/linux/ipc_namespace.h | 13 ++++++++--

```

> ipc/msg.c          | 26 ++++-----
> ipc/namespace.c    | 25 ++++-----
> ipc/sem.c          | 26 ++++-----
> ipc/shm.c          | 26 ++++-----
> ipc/util.c         | 6 +++---
> ipc/util.h         | 16 ++++-----
> 7 files changed, 34 insertions(+), 104 deletions(-)
>
> Index: b/include/linux/ipc_namespace.h
> =====
> --- a/include/linux/ipc_namespace.h
> +++ b/include/linux/ipc_namespace.h
> @@ -2,11 +2,20 @@
> #define __IPC_NAMESPACE_H__
>
> #include <linux/err.h>
> +#include <linux/idr.h>
> +#include <linux/rwsem.h>
> +
> +struct ipc_ids {
> + int in_use;
> + unsigned short seq;
> + unsigned short seq_max;
> + struct rw_semaphore rw_mutex;
> + struct idr ipcs_idr;
> +};
>
> -struct ipc_ids;
> struct ipc_namespace {
> struct kref kref;
> - struct ipc_ids *ids[3];
> + struct ipc_ids ids[3];
>
> int sem_ctls[4];
> int used_sems;
> Index: b/ipc/msg.c
> =====
> --- a/ipc/msg.c
> +++ b/ipc/msg.c
> @@ -67,9 +67,7 @@ struct msg_sender {
> #define SEARCH_NOTEQUAL 3
> #define SEARCH_LESSEQUAL 4
>
> -static struct ipc_ids init_msg_ids;
> -
> -#define msg_ids(ns) (*((ns)->ids[IPC_MSG_IDS]))
> +#define msg_ids(ns) ((ns)->ids[IPC_MSG_IDS])
>

```

```

> #define msg_unlock(msq) ipc_unlock(&(msq)->q_perm)
> #define msg_buidid(id, seq) ipc_buidid(id, seq)
> @@ -80,30 +78,17 @@ static int newque(struct ipc_namespace *
> static int sysvipc_msg_proc_show(struct seq_file *s, void *it);
> #endif
>
> -static void __msg_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)
> +void msg_init_ns(struct ipc_namespace *ns)
> {
> - ns->ids[IPC_MSG_IDS] = ids;
>   ns->msg_ctlmax = MSGMAX;
>   ns->msg_ctlmnb = MSGMNB;
>   ns->msg_ctlmni = MSGMNI;
>   atomic_set(&ns->msg_bytes, 0);
>   atomic_set(&ns->msg_hdrs, 0);
> - ipc_init_ids(ids);
> + ipc_init_ids(&ns->ids[IPC_MSG_IDS]);
> }
>
> #ifdef CONFIG_IPC_NS
> -int msg_init_ns(struct ipc_namespace *ns)
> -{
> - struct ipc_ids *ids;
> -
> - ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
> - if (ids == NULL)
> - return -ENOMEM;
> -
> - __msg_init_ns(ns, ids);
> - return 0;
> -}
> -
> void msg_exit_ns(struct ipc_namespace *ns)
> {
>   struct msg_queue *msq;
> @@ -126,15 +111,12 @@ void msg_exit_ns(struct ipc_namespace *n
> }
>
> up_write(&msg_ids(ns).rw_mutex);
> -
> - kfree(ns->ids[IPC_MSG_IDS]);
> - ns->ids[IPC_MSG_IDS] = NULL;
> }
> #endif
>
> void __init msg_init(void)
> {
> - __msg_init_ns(&init_ipc_ns, &init_msg_ids);

```

```

> + msg_init_ns(&init_ipc_ns);
> ipc_init_proc_interface("sysvipc/msg",
> "      key      msqid perms      cbytes      qnum lspid lrpid  uid  gid  cuid  cgid      stime
rtime      ctime\n",
> IPC_MSG_IDS, sysvipc_msg_proc_show);
> Index: b/ipc/namespace.c
> =====
> --- a/ipc/namespace.c
> +++ b/ipc/namespace.c
> @@ -14,35 +14,18 @@
>
> static struct ipc_namespace *clone_ipc_ns(struct ipc_namespace *old_ns)
> {
> - int err;
> - struct ipc_namespace *ns;
>
> - err = -ENOMEM;
> - ns = kmalloc(sizeof(struct ipc_namespace), GFP_KERNEL);
> - if (ns == NULL)
> - goto err_mem;
> + return ERR_PTR(-ENOMEM);
>
> - err = sem_init_ns(ns);
> - if (err)
> - goto err_sem;
> - err = msg_init_ns(ns);
> - if (err)
> - goto err_msg;
> - err = shm_init_ns(ns);
> - if (err)
> - goto err_shm;
> + sem_init_ns(ns);
> + msg_init_ns(ns);
> + shm_init_ns(ns);
>
> - kref_init(&ns->kref);
> - return ns;
> -
> -err_shm:
> - msg_exit_ns(ns);
> -err_msg:
> - sem_exit_ns(ns);
> -err_sem:
> - kfree(ns);
> -err_mem:
> - return ERR_PTR(err);
> }
>

```

```

> struct ipc_namespace *copy_ipcs(unsigned long flags, struct ipc_namespace *ns)
> Index: b/ipc/sem.c
> =====
> --- a/ipc/sem.c
> +++ b/ipc/sem.c
> @@ -87,14 +87,12 @@
> #include <asm/uaccess.h>
> #include "util.h"
>
> #define sem_ids(ns) (*((ns)->ids[IPC_SEM_IDS]))
> +#define sem_ids(ns) ((ns)->ids[IPC_SEM_IDS])
>
> #define sem_unlock(sma) ipc_unlock(&(sma)->sem_perm)
> #define sem_checkid(sma, semid) ipc_checkid(&sma->sem_perm, semid)
> #define sem_buildid(id, seq) ipc_buildid(id, seq)
>
> -static struct ipc_ids init_sem_ids;
> -
> static int newary(struct ipc_namespace *, struct ipc_params *);
> static void freeary(struct ipc_namespace *, struct sem_array *);
> #ifdef CONFIG_PROC_FS
> @@ -118,30 +116,17 @@ static int sysvipc_sem_proc_show(struct
> #define sc_semopm sem_ctls[2]
> #define sc_semmni sem_ctls[3]
>
> -static void __sem_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)
> +void sem_init_ns(struct ipc_namespace *ns)
> {
> - ns->ids[IPC_SEM_IDS] = ids;
> ns->sc_semmsl = SEMMSL;
> ns->sc_semmns = SEMMNS;
> ns->sc_semopm = SEMOPM;
> ns->sc_semmni = SEMMNI;
> ns->used_sems = 0;
> - ipc_init_ids(ids);
> + ipc_init_ids(&ns->ids[IPC_SEM_IDS]);
> }
>
> #ifdef CONFIG_IPC_NS
> -int sem_init_ns(struct ipc_namespace *ns)
> {-{
> - struct ipc_ids *ids;
> -
> - ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
> - if (ids == NULL)
> - return -ENOMEM;
> -
> - __sem_init_ns(ns, ids);

```

```

> - return 0;
> -}
> -
> void sem_exit_ns(struct ipc_namespace *ns)
> {
>   struct sem_array *sma;
>   @@ -163,15 +148,12 @@ void sem_exit_ns(struct ipc_namespace *n
>   total++;
> }
>   up_write(&sem_ids(ns).rw_mutex);
> -
> - kfree(ns->ids[IPC_SEM_IDS]);
> - ns->ids[IPC_SEM_IDS] = NULL;
> }
> #endif
>
> void __init sem_init (void)
> {
> - __sem_init_ns(&init_ipc_ns, &init_sem_ids);
> + sem_init_ns(&init_ipc_ns);
>   ipc_init_proc_interface("sysvipc/sem",
>   "      key      semid perms   nsems uid  gid  cuid  cgid   otime   ctime\n",
>   IPC_SEM_IDS, sysvipc_sem_proc_show);
> Index: b/ipc/util.c
> =====
> --- a/ipc/util.c
> +++ b/ipc/util.c
> @@ -782,7 +782,7 @@ static void *sysvipc_proc_next(struct se
>   if (ipc && ipc != SEQ_START_TOKEN)
>     ipc_unlock(ipc);
>
> - return sysvipc_find_ipc(iter->ns->ids[iface->ids], *pos, pos);
> + return sysvipc_find_ipc(&iter->ns->ids[iface->ids], *pos, pos);
> }
>
> /*
> @@ -795,7 +795,7 @@ static void *sysvipc_proc_start(struct s
>   struct ipc_proc_iface *iface = iter->iface;
>   struct ipc_ids *ids;
>
> - ids = iter->ns->ids[iface->ids];
> + ids = &iter->ns->ids[iface->ids];
>
> /*
>   * Take the lock - this will be released by the corresponding
> @@ -826,7 +826,7 @@ static void sysvipc_proc_stop(struct seq
>   if (ipc && ipc != SEQ_START_TOKEN)
>     ipc_unlock(ipc);

```

```

>
> - ids = iter->ns->ids[iface->ids];
> + ids = &iter->ns->ids[iface->ids];
> /* Release the lock we took in start() */
> up_read(&ids->rw_mutex);
> }
> Index: b/ipc/util.h
> =====
> --- a/ipc/util.h
> +++ b/ipc/util.h
> @@ -10,7 +10,6 @@
> #ifndef _IPC_UTIL_H
> #define _IPC_UTIL_H
>
> #include <linux/idr.h>
> #include <linux/err.h>
>
> #define USHRT_MAX 0xffff
> @@ -22,22 +21,14 @@ void shm_init (void);
>
> struct ipc_namespace;
>
> -int sem_init_ns(struct ipc_namespace *ns);
> -int msg_init_ns(struct ipc_namespace *ns);
> -int shm_init_ns(struct ipc_namespace *ns);
> +void sem_init_ns(struct ipc_namespace *ns);
> +void msg_init_ns(struct ipc_namespace *ns);
> +void shm_init_ns(struct ipc_namespace *ns);
>
> void sem_exit_ns(struct ipc_namespace *ns);
> void msg_exit_ns(struct ipc_namespace *ns);
> void shm_exit_ns(struct ipc_namespace *ns);
>
> -struct ipc_ids {
> - int in_use;
> - unsigned short seq;
> - unsigned short seq_max;
> - struct rw_semaphore rw_mutex;
> - struct idr ipcs_idr;
> -};
> -
> /*
>  * Structure that holds the parameters needed by the ipc operations
>  * (see after)
> @@ -68,6 +59,7 @@ struct ipc_ops {
> };
>
> struct seq_file;

```

```

> +struct ipc_ids;
>
> void ipc_init_ids(struct ipc_ids *);
> #ifdef CONFIG_PROC_FS
> Index: b/ipc/shm.c
> =====
> --- a/ipc/shm.c
> +++ b/ipc/shm.c
> @@ -56,9 +56,7 @@ struct shm_file_data {
> static const struct file_operations shm_file_operations;
> static struct vm_operations_struct shm_vm_ops;
>
> -static struct ipc_ids init_shm_ids;
> -
> -#define shm_ids(ns) (*((ns)->ids[IPC_SHM_IDS]))
> +#define shm_ids(ns) ((ns)->ids[IPC_SHM_IDS])
>
> #define shm_unlock(shp) \
> ipc_unlock(&(shp)->shm_perm)
> @@ -72,14 +70,13 @@ static void shm_destroy (struct ipc_name
> static int sysvipc_shm_proc_show(struct seq_file *s, void *it);
> #endif
>
> -static void __shm_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)
> +void shm_init_ns(struct ipc_namespace *ns)
> {
> - ns->ids[IPC_SHM_IDS] = ids;
> ns->shm_ctlmax = SHMMAX;
> ns->shm_ctlall = SHMALL;
> ns->shm_ctlmni = SHMMNI;
> ns->shm_tot = 0;
> - ipc_init_ids(ids);
> + ipc_init_ids(&ns->ids[IPC_SHM_IDS]);
> }
>
> /*
> @@ -98,18 +95,6 @@ static void do_shm_rmid(struct ipc_names
> }
>
> #ifdef CONFIG_IPC_NS
> -int shm_init_ns(struct ipc_namespace *ns)
> -{
> - struct ipc_ids *ids;
> -
> - ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
> - if (ids == NULL)
> - return -ENOMEM;
> -

```

```

> - __shm_init_ns(ns, ids);
> - return 0;
> -}
> -
> void shm_exit_ns(struct ipc_namespace *ns)
> {
> struct shmid_kernel *shp;
> @@ -131,15 +116,12 @@ void shm_exit_ns(struct ipc_namespace *n
> total++;
> }
> up_write(&shm_ids(ns).rw_mutex);
> -
> - kfree(ns->ids[IPC_SHM_IDS]);
> - ns->ids[IPC_SHM_IDS] = NULL;
> }
> #endif
>
> void __init shm_init (void)
> {
> - __shm_init_ns(&init_ipc_ns, &init_shm_ids);
> + shm_init_ns(&init_ipc_ns);
> ipc_init_proc_interface("sysvipc/shm",
> "      key      shmid perms      size cpid lpid nattch  uid  gid cuid cgid      atime      dtime
ctime\n",
> IPC_SHM_IDS, sysvipc_shm_proc_show);
>

```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: make struct ipc_ids static in ipc_namespace

Posted by [Pavel Emelianov](#) on Fri, 23 Nov 2007 07:37:43 GMT

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

Cedric Le Goater wrote:

> Pierre Peiffer wrote:

>> Each ipc_namespace contains a table of 3 pointers to struct ipc_ids (3 for

>> msg, sem and shm, structure used to store all ipcs)

>> These 'struct ipc_ids' are dynamically allocated for each ipc_namespace as

>> the ipc_namespace itself (for the init namespace, they are initialized with

>> pointers to static variables instead)

>>

>> It is so for historical reason: in fact, before the use of idr to store the

>> ipcs, the ipcs were stored in tables of variable length, depending of the

```

>> maximum number of ipc allowed.
>> Now, these 'struct ipc_ids' have a fixed size. As they are allocated in any
>> cases for each new ipc_namespace, there is no gain of memory in having them
>> allocated separately of the struct ipc_namespace.
>>
>> This patch proposes to make this table static in the struct ipc_namespace.
>> Thus, we can allocate all in once and get rid of all the code needed to
>> allocate and free these ipc_ids separately.
>
> It looks safe and saves quite a lot of line.
>
> Pavel, what do you think of it ?

```

Looks sane, good catch, Pierre.

But I'd find out whether these three ipc_ids intersect any cache-line. In other words I'd mark the struct ipc_ids as ____cacheline_aligned and checked for any differences.

```

> Acked-by: Cedric Le Goater <clg@fr.ibm.com>
>
> Thanks,

```

Thanks,
Pavel

```

> C.
>
>
>> Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>
>> ---
>> include/linux/ipc_namespace.h | 13 ++++++++--
>> ipc/msg.c                     | 26 +++++-----
>> ipc/namespace.c               | 25 +++++-----
>> ipc/sem.c                     | 26 +++++-----
>> ipc/shm.c                     | 26 +++++-----
>> ipc/util.c                    |  6 +---
>> ipc/util.h                    | 16 +++++-----
>> 7 files changed, 34 insertions(+), 104 deletions(-)
>>
>> Index: b/include/linux/ipc_namespace.h
>> =====
>> --- a/include/linux/ipc_namespace.h
>> +++ b/include/linux/ipc_namespace.h
>> @@ -2,11 +2,20 @@
>> #define __IPC_NAMESPACE_H__
>>
>> #include <linux/err.h>

```

```

>> + #include <linux/idr.h>
>> + #include <linux/rwsem.h>
>> +
>> + struct ipc_ids {
>> + int in_use;
>> + unsigned short seq;
>> + unsigned short seq_max;
>> + struct rw_semaphore rw_mutex;
>> + struct idr ipcs_idr;
>> +};
>>
>> - struct ipc_ids;
>> struct ipc_namespace {
>> struct kref kref;
>> - struct ipc_ids *ids[3];
>> + struct ipc_ids ids[3];
>>
>> int sem_ctls[4];
>> int used_sems;
>> Index: b/ipc/msg.c
>> =====
>> --- a/ipc/msg.c
>> +++ b/ipc/msg.c
>> @@ -67,9 +67,7 @@ struct msg_sender {
>> #define SEARCH_NOTEQUAL 3
>> #define SEARCH_LESSEQUAL 4
>>
>> -static struct ipc_ids init_msg_ids;
>> -
>> -#define msg_ids(ns) ((ns)->ids[IPC_MSG_IDS])
>> + #define msg_ids(ns) ((ns)->ids[IPC_MSG_IDS])
>>
>> #define msg_unlock(msq) ipc_unlock(&(msq)->q_perm)
>> #define msg_buildid(id, seq) ipc_buildid(id, seq)
>> @@ -80,30 +78,17 @@ static int newque(struct ipc_namespace *
>> static int sysvipc_msg_proc_show(struct seq_file *s, void *it);
>> #endif
>>
>> -static void __msg_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)
>> +void msg_init_ns(struct ipc_namespace *ns)
>> {
>> - ns->ids[IPC_MSG_IDS] = ids;
>> ns->msg_ctlmax = MSGMAX;
>> ns->msg_ctlmnb = MSGMNB;
>> ns->msg_ctlmni = MSGMNI;
>> atomic_set(&ns->msg_bytes, 0);
>> atomic_set(&ns->msg_hdrs, 0);
>> - ipc_init_ids(ids);

```

```

>> + ipc_init_ids(&ns->ids[IPC_MSG_IDS]);
>> }
>>
>> #ifdef CONFIG_IPC_NS
>> -int msg_init_ns(struct ipc_namespace *ns)
>> -{
>> - struct ipc_ids *ids;
>> -
>> - ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
>> - if (ids == NULL)
>> - return -ENOMEM;
>> -
>> - __msg_init_ns(ns, ids);
>> - return 0;
>> -}
>> -
>> void msg_exit_ns(struct ipc_namespace *ns)
>> {
>> struct msg_queue *msgq;
>> @@ -126,15 +111,12 @@ void msg_exit_ns(struct ipc_namespace *n
>> }
>>
>> up_write(&msg_ids(ns).rw_mutex);
>> -
>> - kfree(ns->ids[IPC_MSG_IDS]);
>> - ns->ids[IPC_MSG_IDS] = NULL;
>> }
>> #endif
>>
>> void __init msg_init(void)
>> {
>> - __msg_init_ns(&init_ipc_ns, &init_msg_ids);
>> + msg_init_ns(&init_ipc_ns);
>> ipc_init_proc_interface("sysvipc/msg",
>> "      key      msgqid perms      cbytes      qnum lspid lrpid  uid   gid   cuid   cgid   stime
>> rtime      ctime\n",
>> IPC_MSG_IDS, sysvipc_msg_proc_show);
>> Index: b/ipc/namespace.c
>> =====
>> --- a/ipc/namespace.c
>> +++ b/ipc/namespace.c
>> @@ -14,35 +14,18 @@
>>
>> static struct ipc_namespace *clone_ipc_ns(struct ipc_namespace *old_ns)
>> {
>> - int err;
>> struct ipc_namespace *ns;
>>

```

```

>> - err = -ENOMEM;
>> ns = kmalloc(sizeof(struct ipc_namespace), GFP_KERNEL);
>> if (ns == NULL)
>> - goto err_mem;
>> + return ERR_PTR(-ENOMEM);
>>
>> - err = sem_init_ns(ns);
>> - if (err)
>> - goto err_sem;
>> - err = msg_init_ns(ns);
>> - if (err)
>> - goto err_msg;
>> - err = shm_init_ns(ns);
>> - if (err)
>> - goto err_shm;
>> + sem_init_ns(ns);
>> + msg_init_ns(ns);
>> + shm_init_ns(ns);
>>
>> kref_init(&ns->kref);
>> return ns;
>> -
>> -err_shm:
>> - msg_exit_ns(ns);
>> -err_msg:
>> - sem_exit_ns(ns);
>> -err_sem:
>> - kfree(ns);
>> -err_mem:
>> - return ERR_PTR(err);
>> }
>>
>> struct ipc_namespace *copy_ipcs(unsigned long flags, struct ipc_namespace *ns)
>> Index: b/ipc/sem.c
>> =====
>> --- a/ipc/sem.c
>> +++ b/ipc/sem.c
>> @@ -87,14 +87,12 @@
>> #include <asm/uaccess.h>
>> #include "util.h"
>>
>> #define sem_ids(ns) *((ns)->ids[IPC_SEM_IDS])
>> + #define sem_ids(ns) ((ns)->ids[IPC_SEM_IDS])
>>
>> #define sem_unlock(sma) ipc_unlock(&(sma)->sem_perm)
>> #define sem_checkid(sma, semid) ipc_checkid(&sma->sem_perm, semid)
>> #define sem_buildid(id, seq) ipc_buildid(id, seq)
>>

```

```

>> -static struct ipc_ids init_sem_ids;
>> -
>> static int newary(struct ipc_namespace *, struct ipc_params *);
>> static void freeary(struct ipc_namespace *, struct sem_array *);
>> #ifdef CONFIG_PROC_FS
>> @@ -118,30 +116,17 @@ static int sysvipc_sem_proc_show(struct
>> #define sc_semopm sem_ctls[2]
>> #define sc_semmni sem_ctls[3]
>>
>> -static void __sem_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)
>> +void sem_init_ns(struct ipc_namespace *ns)
>> {
>> - ns->ids[IPC_SEM_IDS] = ids;
>> ns->sc_semmnl = SEMMNL;
>> ns->sc_semmns = SEMMNS;
>> ns->sc_semopm = SEMOPM;
>> ns->sc_semmni = SEMMNI;
>> ns->used_sems = 0;
>> - ipc_init_ids(ids);
>> + ipc_init_ids(&ns->ids[IPC_SEM_IDS]);
>> }
>>
>> #ifdef CONFIG_IPC_NS
>> -int sem_init_ns(struct ipc_namespace *ns)
>> -{
>> - struct ipc_ids *ids;
>> -
>> - ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
>> - if (ids == NULL)
>> - return -ENOMEM;
>> -
>> - __sem_init_ns(ns, ids);
>> - return 0;
>> -}
>> -
>> void sem_exit_ns(struct ipc_namespace *ns)
>> {
>> struct sem_array *sma;
>> @@ -163,15 +148,12 @@ void sem_exit_ns(struct ipc_namespace *n
>> total++;
>> }
>> up_write(&sem_ids(ns).rw_mutex);
>> -
>> - kfree(ns->ids[IPC_SEM_IDS]);
>> - ns->ids[IPC_SEM_IDS] = NULL;
>> }
>> #endif
>>

```

```

>> void __init sem_init (void)
>> {
>> - __sem_init_ns(&init_ipc_ns, &init_sem_ids);
>> + sem_init_ns(&init_ipc_ns);
>> ipc_init_proc_interface("sysvipc/sem",
>>     "      key      semid perms   nsems uid  gid cuid  cgid   otime   ctime\n",
>>     IPC_SEM_IDS, sysvipc_sem_proc_show);
>> Index: b/ipc/util.c
>> =====
>> --- a/ipc/util.c
>> +++ b/ipc/util.c
>> @@ -782,7 +782,7 @@ static void *sysvipc_proc_next(struct se
>> if (ipc && ipc != SEQ_START_TOKEN)
>>     ipc_unlock(ipc);
>>
>> - return sysvipc_find_ipc(iter->ns->ids[iface->ids], *pos, pos);
>> + return sysvipc_find_ipc(&iter->ns->ids[iface->ids], *pos, pos);
>> }
>>
>> /*
>> @@ -795,7 +795,7 @@ static void *sysvipc_proc_start(struct s
>> struct ipc_proc_iface *iface = iter->iface;
>> struct ipc_ids *ids;
>>
>> - ids = iter->ns->ids[iface->ids];
>> + ids = &iter->ns->ids[iface->ids];
>>
>> /*
>>  * Take the lock - this will be released by the corresponding
>> @@ -826,7 +826,7 @@ static void sysvipc_proc_stop(struct seq
>> if (ipc && ipc != SEQ_START_TOKEN)
>>     ipc_unlock(ipc);
>>
>> - ids = iter->ns->ids[iface->ids];
>> + ids = &iter->ns->ids[iface->ids];
>> /* Release the lock we took in start() */
>> up_read(&ids->rw_mutex);
>> }
>> Index: b/ipc/util.h
>> =====
>> --- a/ipc/util.h
>> +++ b/ipc/util.h
>> @@ -10,7 +10,6 @@
>> #ifndef _IPC_UTIL_H
>> #define _IPC_UTIL_H
>>
>> #include <linux/idr.h>
>> #include <linux/err.h>

```

```

>>
>> #define USHRT_MAX 0xffff
>> @@ -22,22 +21,14 @@ void shm_init (void);
>>
>> struct ipc_namespace;
>>
>> -int sem_init_ns(struct ipc_namespace *ns);
>> -int msg_init_ns(struct ipc_namespace *ns);
>> -int shm_init_ns(struct ipc_namespace *ns);
>> +void sem_init_ns(struct ipc_namespace *ns);
>> +void msg_init_ns(struct ipc_namespace *ns);
>> +void shm_init_ns(struct ipc_namespace *ns);
>>
>> void sem_exit_ns(struct ipc_namespace *ns);
>> void msg_exit_ns(struct ipc_namespace *ns);
>> void shm_exit_ns(struct ipc_namespace *ns);
>>
>> -struct ipc_ids {
>> - int in_use;
>> - unsigned short seq;
>> - unsigned short seq_max;
>> - struct rw_semaphore rw_mutex;
>> - struct idr ipcs_idr;
>> -};
>> -
>> /*
>>  * Structure that holds the parameters needed by the ipc operations
>>  * (see after)
>> @@ -68,6 +59,7 @@ struct ipc_ops {
>> };
>>
>> struct seq_file;
>> +struct ipc_ids;
>>
>> void ipc_init_ids(struct ipc_ids *);
>> #ifdef CONFIG_PROC_FS
>> Index: b/ipc/shm.c
>> =====
>> --- a/ipc/shm.c
>> +++ b/ipc/shm.c
>> @@ -56,9 +56,7 @@ struct shm_file_data {
>> static const struct file_operations shm_file_operations;
>> static struct vm_operations_struct shm_vm_ops;
>>
>> -static struct ipc_ids init_shm_ids;
>> -
>> -#define shm_ids(ns) (*((ns)->ids[IPC_SHM_IDS]))
>> +#define shm_ids(ns) ((ns)->ids[IPC_SHM_IDS])

```

```

>>
>> #define shm_unlock(shp) \
>> ipc_unlock(&(shp)->shm_perm)
>> @@ -72,14 +70,13 @@ static void shm_destroy (struct ipc_name
>> static int sysvipc_shm_proc_show(struct seq_file *s, void *it);
>> #endif
>>
>> -static void __shm_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)
>> +void shm_init_ns(struct ipc_namespace *ns)
>> {
>> - ns->ids[IPC_SHM_IDS] = ids;
>> ns->shm_ctlmax = SHMMAX;
>> ns->shm_ctlall = SHMALL;
>> ns->shm_ctlmni = SHMMNI;
>> ns->shm_tot = 0;
>> - ipc_init_ids(ids);
>> + ipc_init_ids(&ns->ids[IPC_SHM_IDS]);
>> }
>>
>> /*
>> @@ -98,18 +95,6 @@ static void do_shm_rmid(struct ipc_names
>> }
>>
>> #ifdef CONFIG_IPC_NS
>> -int shm_init_ns(struct ipc_namespace *ns)
>> -{
>> - struct ipc_ids *ids;
>> -
>> - ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
>> - if (ids == NULL)
>> - return -ENOMEM;
>> -
>> - __shm_init_ns(ns, ids);
>> - return 0;
>> -}
>> -
>> void shm_exit_ns(struct ipc_namespace *ns)
>> {
>> struct shmid_kernel *shp;
>> @@ -131,15 +116,12 @@ void shm_exit_ns(struct ipc_namespace *n
>> total++;
>> }
>> up_write(&shm_ids(ns).rw_mutex);
>> -
>> - kfree(ns->ids[IPC_SHM_IDS]);
>> - ns->ids[IPC_SHM_IDS] = NULL;
>> }
>> #endif

```

```
>>
>> void __init shm_init (void)
>> {
>> - __shm_init_ns(&init_ipc_ns, &init_shm_ids);
>> + shm_init_ns(&init_ipc_ns);
>> ipc_init_proc_interface("sysvipc/shm",
>>   "      key      shmids perms      size cpid lpid nattch  uid  gid cuid cgid   atime   dtime
      ctime\n",
>>   IPC_SHM_IDS, sysvipc_shm_proc_show);
>>
>
>
```

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

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: make struct ipc_ids static in
ipc_namespace

Posted by [Pavel Emelianov](#) on Fri, 23 Nov 2007 08:02:47 GMT

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

Pavel Emelyanov wrote:

> Cedric Le Goater wrote:

>> Pierre Peiffer wrote:

[snip]

>> Pavel, what do you think of it ?

>

> Looks sane, good catch, Pierre.

>

> But I'd find out whether these three ipc_ids intersect any

> cache-line. In other words I'd mark the struct ipc_ids as

> ____cacheline_aligned and checked for any differences.

BTW! It might be also useful to keep ipc_ids closer to their
sysctl parameters.

>> Acked-by: Cedric Le Goater <clg@fr.ibm.com>

>>

>> Thanks,

>

> Thanks,

> Pavel

>

>> C.

[snip]

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

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: make struct ipc_ids static in ipc_namespace

Posted by [Pierre Peiffer](#) on Fri, 23 Nov 2007 08:58:24 GMT

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

Hi,

Thanks for reviewing this !

Pavel Emelyanov wrote:

> Pavel Emelyanov wrote:

>> Cedric Le Goater wrote:

>>> Pierre Peiffer wrote:

>

> [snip]

>

>>> Pavel, what do you think of it ?

>> Looks sane, good catch, Pierre.

>>

>> But I'd find out whether these three ipc_ids intersect any

>> cache-line. In other words I'd mark the struct ipc_ids as

>> ____cacheline_aligned and checked for any differences.

>

> BTW! It might be also useful to keep ipc_ids closer to their

> sysctl parameters.

>

It makes sense indeed.

That would mean to have something like this, right ?

```
struct ipc_namespace {  
    struct kref kref;
```

```
  
    struct ipc_ids sem_ids;  
    int  sem_ctls[4];  
    int  used_sems;
```

```
  
    struct ipc_ids msg_ids;
```

```
int msg_ctlmax;
int msg_ctlmnb;
int msg_ctlmni;
atomic_t msg_bytes;
atomic_t msg_hdrs;

struct ipc_ids shm_ids;
size_t shm_ctlmax;
size_t shm_ctlall;
int shm_ctlmni;
int shm_tot;
};
```

After a quick look, that implies to rework a little bit procfs... othwise, it's not a big deal as I can see.

P.

```
>>> Acked-by: Cedric Le Goater <clg@fr.ibm.com>
>>>
>>> Thanks,
>> Thanks,
>> Pavel
>>
>>> C.
>
> [snip]
> -
> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html
> Please read the FAQ at http://www.tux.org/lkml/
>
```

--
Pierre Peiffer

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

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: make struct ipc_ids static in ipc_namespace
Posted by [Pavel Emelianov](#) on Fri, 23 Nov 2007 09:27:15 GMT
[View Forum Message](#) <> [Reply to Message](#)

Pierre Peiffer wrote:

> Hi,
>
> Thanks for reviewing this !
>
> Pavel Emelyanov wrote:
>> Pavel Emelyanov wrote:
>>> Cedric Le Goater wrote:
>>>> Pierre Peiffer wrote:
>> [snip]
>>
>>>> Pavel, what do you think of it ?
>>> Looks sane, good catch, Pierre.
>>>
>>> But I'd find out whether these three ipc_ids intersect any
>>> cache-line. In other words I'd mark the struct ipc_ids as
>>> ____cacheline_aligned and checked for any differences.
>> BTW! It might be also useful to keep ipc_ids closer to their
>> sysctl parameters.
>>
>
> It makes sense indeed.
>
> That would mean to have something like this, right ?

Yup :)

```
> struct ipc_namespace {  
> struct kref kref;  
>  
> struct ipc_ids sem_ids;  
> int sem_ctls[4];  
> int used_sems;  
>  
> struct ipc_ids msg_ids;  
> int msg_ctlmax;  
> int msg_ctlmnb;  
> int msg_ctlmni;  
> atomic_t msg_bytes;  
> atomic_t msg_hdrs;  
>  
> struct ipc_ids shm_ids;  
> size_t shm_ctlmax;  
> size_t shm_ctlall;  
> int shm_ctlmni;  
> int shm_tot;  
> };  
>  
> After a quick look, that implies to rework a little bit procfs... othwise, it's
```

> not a big deal as I can see.

Thanks!

> P.

>

>>>> Acked-by: Cedric Le Goater <clg@fr.ibm.com>

>>>>

>>>> Thanks,

>>> Thanks,

>>> Pavel

>>>

>>>> C.

>> [snip]

>> -

>> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in

>> the body of a message to majordomo@vger.kernel.org

>> More majordomo info at <http://vger.kernel.org/majordomo-info.html>

>> Please read the FAQ at <http://www.tux.org/lkml/>

>>

>

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: make struct ipc_ids static in ipc_namespace

Posted by [Pierre Peiffer](#) on Fri, 23 Nov 2007 10:49:11 GMT

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

Ok, I have the patch ready, but before sending it, I worry about the size of struct ipc_namespace if we mark struct ipc_ids as __cacheline_aligned....

Of course, you we fall into a classical match: performance vs memory size.

As I don't think that I have the knowledge to decide what we must focus on, here after is, for info, the size reported by pahole (on x86, Intel Xeon)

With the patch sent at the beginning of this thread we have:

```
struct ipc_namespace {
    struct kref          kref;          /* 0 4 */
    struct ipc_ids       ids[3];        /* 4 156 */
    /* --- cacheline 2 boundary (128 bytes) was 32 bytes ago --- */
    int                  sem_ctls[4];    /* 160 16 */
}
```

```

int            used_sems;        /* 176  4 */
int            msg_ctlmax;       /* 180  4 */
int            msg_ctlmnb;       /* 184  4 */
int            msg_ctlmni;       /* 188  4 */
/* --- cacheline 3 boundary (192 bytes) --- */
atomic_t       msg_bytes;        /* 192  4 */
atomic_t       msg_hdrs;         /* 196  4 */
size_t         shm_ctlmax;       /* 200  4 */
size_t         shm_ctlall;       /* 204  4 */
int            shm_ctlmni;       /* 208  4 */
int            shm_tot;          /* 212  4 */

/* size: 216, cachelines: 4 */
/* last cacheline: 24 bytes */
}; /* definitions: 1 */

```

With the new patch, if we mark the struct ipc_ids as `____cacheline_aligned`, we have (I put kref at the end, to save one more cacheline):

```

struct ipc_namespace {
    struct ipc_ids    sem_ids;      /* 0  64 */

    /* XXX last struct has 12 bytes of padding */

    /* --- cacheline 1 boundary (64 bytes) --- */
    int               sem_ctls[4];  /* 64 16 */
    int               used_sems;    /* 80  4 */

    /* XXX 44 bytes hole, try to pack */

    /* --- cacheline 2 boundary (128 bytes) --- */
    struct ipc_ids    msg_ids;      /* 128 64 */

    /* XXX last struct has 12 bytes of padding */

    /* --- cacheline 3 boundary (192 bytes) --- */
    int               msg_ctlmax;    /* 192  4 */
    int               msg_ctlmnb;    /* 196  4 */
    int               msg_ctlmni;    /* 200  4 */
    atomic_t          msg_bytes;     /* 204  4 */
    atomic_t          msg_hdrs;     /* 208  4 */

    /* XXX 44 bytes hole, try to pack */

    /* --- cacheline 4 boundary (256 bytes) --- */
    struct ipc_ids    shm_ids;      /* 256 64 */

    /* XXX last struct has 12 bytes of padding */
}

```

```

/* --- cacheline 5 boundary (320 bytes) --- */
size_t      shm_ctlmax;      /* 320  4 */
size_t      shm_ctlall;      /* 324  4 */
int         shm_ctlmni;      /* 328  4 */
int         shm_tot;         /* 332  4 */
struct kref  kref;           /* 336  4 */

/* size: 384, cachelines: 6 */
/* sum members: 252, holes: 2, sum holes: 88 */
/* padding: 44 */
/* paddings: 3, sum paddings: 36 */
}; /* definitions: 1 */

```

We can put all sysctl related values together, in one cacheline and keep ipc_ids cacheline aligned ? But I really wonder about the performance gain here...

Humm humm, comment ?

P.

Pavel Emelyanov wrote:

> Pierre Peiffer wrote:

>> Hi,

>>

>> Thanks for reviewing this !

>>

>> Pavel Emelyanov wrote:

>>> Pavel Emelyanov wrote:

>>>> Cedric Le Goater wrote:

>>>>> Pierre Peiffer wrote:

>>> [snip]

>>>

>>>>> Pavel, what do you think of it ?

>>>> Looks sane, good catch, Pierre.

>>>>

>>>> But I'd find out whether these three ipc_ids intersect any

>>>> cache-line. In other words I'd mark the struct ipc_ids as

>>>> ____cacheline_aligned and checked for any differences.

>>> BTW! It might be also useful to keep ipc_ids closer to their

>>> sysctl parameters.

>>>

>> It makes sense indeed.

>>

>> That would mean to have something like this, right ?

>

> Yup :)

>

```

>> struct ipc_namespace {
>> struct kref kref;
>>
>> struct ipc_ids sem_ids;
>> int sem_ctls[4];
>> int used_sems;
>>
>> struct ipc_ids msg_ids;
>> int msg_ctlmax;
>> int msg_ctlmnb;
>> int msg_ctlmni;
>> atomic_t msg_bytes;
>> atomic_t msg_hdrs;
>>
>> struct ipc_ids shm_ids;
>> size_t shm_ctlmax;
>> size_t shm_ctlall;
>> int shm_ctlmni;
>> int shm_tot;
>> };
>>
>> After a quick look, that implies to rework a little bit procfs... othwise, it's
>> not a big deal as I can see.
>
> Thanks!
>
>> P.
>>
>>>>> Acked-by: Cedric Le Goater <clg@fr.ibm.com>
>>>>>
>>>>> Thanks,
>>>> Thanks,
>>>> Pavel
>>>>
>>>>> C.
>>> [snip]
>>> -
>>> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
>>> the body of a message to majordomo@vger.kernel.org
>>> More majordomo info at http://vger.kernel.org/majordomo-info.html
>>> Please read the FAQ at http://www.tux.org/lkml/
>>>
>
>
--
Pierre Peiffer

```

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: make struct ipc_ids static in ipc_namespace

Posted by [Pavel Emelianov](#) on Fri, 23 Nov 2007 11:06:49 GMT

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

Pierre Peiffer wrote:

> Ok, I have the patch ready, but before sending it, I worry about the size of
> struct ipc_namespace if we mark struct ipc_ids as ____cacheline_aligned....
>
> Of course, you we fall into a classical match: performance vs memory size.
>
> As I don't think that I have the knowledge to decide what we must focus on, here
> after is, for info, the size reported by pahole (on x86, Intel Xeon)
>
> With the patch sent at the beginning of this thread we have:
>

```
> struct ipc_namespace {  
>     struct kref      kref;          /* 0 4 */  
>     struct ipc_ids    ids[3];        /* 4 156 */  
>     /* --- cacheline 2 boundary (128 bytes) was 32 bytes ago --- */  
>     int               sem_ctls[4];    /* 160 16 */  
>     int               used_sems;      /* 176 4 */  
>     int               msg_ctlmax;     /* 180 4 */  
>     int               msg_ctlmnb;     /* 184 4 */  
>     int               msg_ctlmni;     /* 188 4 */  
>     /* --- cacheline 3 boundary (192 bytes) --- */  
>     atomic_t          msg_bytes;      /* 192 4 */  
>     atomic_t          msg_hdrs;       /* 196 4 */
```

cacheline boundary is to be here as well... But anyway, please, see my last comment :)

```
>     size_t            shm_ctlmax;     /* 200 4 */  
>     size_t            shm_ctlall;     /* 204 4 */  
>     int               shm_ctlmni;     /* 208 4 */  
>     int               shm_tot;        /* 212 4 */  
>  
>     /* size: 216, cachelines: 4 */  
>     /* last cacheline: 24 bytes */  
> }; /* definitions: 1 */  
>
```

> With the new patch, if we mark the struct ipc_ids as ____cacheline_aligned, we
> have (I put kref at the end, to save one more cacheline):

```

>
> struct ipc_namespace {
>     struct ipc_ids      sem_ids;          /* 0 64 */
>
>     /* XXX last struct has 12 bytes of padding */
>
>     /* --- cacheline 1 boundary (64 bytes) --- */
>     int      sem_ctls[4];      /* 64 16 */
>     int      used_sems;        /* 80 4 */
>
>     /* XXX 44 bytes hole, try to pack */
>
>     /* --- cacheline 2 boundary (128 bytes) --- */
>     struct ipc_ids      msg_ids;          /* 128 64 */
>
>     /* XXX last struct has 12 bytes of padding */
>
>     /* --- cacheline 3 boundary (192 bytes) --- */
>     int      msg_ctlmax;      /* 192 4 */
>     int      msg_ctlmnb;      /* 196 4 */
>     int      msg_ctlmni;      /* 200 4 */
>     atomic_t  msg_bytes;      /* 204 4 */
>     atomic_t  msg_hdrs;       /* 208 4 */
>
>     /* XXX 44 bytes hole, try to pack */
>
>     /* --- cacheline 4 boundary (256 bytes) --- */
>     struct ipc_ids      shm_ids;          /* 256 64 */
>
>     /* XXX last struct has 12 bytes of padding */
>
>     /* --- cacheline 5 boundary (320 bytes) --- */
>     size_t      shm_ctlmax;      /* 320 4 */
>     size_t      shm_ctlall;      /* 324 4 */
>     int      shm_ctlmni;      /* 328 4 */
>     int      shm_tot;          /* 332 4 */
>     struct kref      kref;        /* 336 4 */
>
>     /* size: 384, cachelines: 6 */
>     /* sum members: 252, holes: 2, sum holes: 88 */
>     /* padding: 44 */
>     /* paddings: 3, sum paddings: 36 */
> }; /* definitions: 1 */
>
> We can put all sysctl related values together, in one cacheline and keep ipc_ids
> cacheline aligned ? But I really wonder about the performance gain...

```

Well I think you're right. The structure gains 50% in size... Really too

much to fight for performance in IPC :)

Thanks for checking this thing.

You may put my Acked-by in the original patch.

Thanks,
Pavel

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

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: make struct ipc_ids static in
ipc_namespace

Posted by [Pierre Peiffer](#) on Fri, 23 Nov 2007 11:40:27 GMT

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

Pavel Emelyanov wrote:

> Well I think you're right. The structure gains 50% in size... Really too

> much to fight for performance in IPC :)

>

> Thanks for checking this thing.

>

> You may put my Acked-by in the original patch.

>

Cool. Thanks !

P.

> Thanks,

> Pavel

>

--

Pierre Peiffer

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