Subject: [PATCH 0/8] Change default MSGMNI tunable to scale with lowmem (v3)
Posted by Nadia Derbey on Mon, 11 Feb 2008 14:16:46 GMT
View Forum Message <> Reply to Message

Resending the set of patches after Yasunori's remark about being able to
turn on/off automatic recomputing.
(see message at http://lkml.org/lkml/2008/2/5/149).
I actually introduced an intermediate solution: when msgmni is set by hand,
it is uneregistered from the ipcns notifier chain (i.e. automatic recomputing
is disabled). This corresponds to an implicit turn off. Setting it to a
negative value makes it registered back in the notifier chain (which
corresponds to the turn on proposed by Yasunaori).

Only patch # 8 introduces new stuff compared to the already sent patch sets.

Also ported the patchset to 2.6.24-mm1.

On large systems we'd like to allow a larger number of message queues.
In some cases up to 32K. However simply setting MSGMNI to a larger value may
cause problems for smaller systems.

The first patch of this series introduces a default maximum number of message
queue ids that scales with the amount of lowmem.

Since msgmni is per namespace and there is no amount of memory dedicated to
each namespace so far, the second patch of this series scales msgmni to
the number of ipc namespaces too.

Since msgmni depends on the amount of memory, it becomes necessary to
recompute it upon memory add/remove.
In the 4th patch, memory hotplug management is added: a notifier block is
registered into the memory hotplug notifier chain for the ipc subsystem.
Since the ipc namespaces are not linked together, they have their own
notification chain: one notifier_block is defined per ipc namespace.
Each time an ipc namespace is created (removed) it registers (unregisters)
its notifier block in (from) the ipcns chain.
The callback routine registered in the memory chain invokes the ipcns
notifier chain with the IPCNS_MEMCHANGE event.
Each callback routine registered in the ipcns namespace, in turn, recomputes
msgmni for the owning namespace.

The 5th patch makes it possible to keep the memory hotplug notifier chain's
lock for a lesser amount of time: instead of directly notifying the ipcns
notifier chain upon memory add/remove, a work item is added to the global
workqueue. When activated, this work item is the one who notifies the ipcns
notifier chain.

Since msgmni depends on the number of ipc namespaces, it becomes necessary
to recompute it upon ipc namespace creation / removal.
The 6th patch uses the ipc namespace notifier chain for that purpose: that
chain is notified each time an ipc namespace is created or removed. This makes
it possible to recompute msgmni for all the namespaces each time one of them
is created or removed.

When msgmni is explicitely set from userspace, we should avoid recomputing it
upon memory add/remove or ipcns creation/removal.
This is what the 7th patch does: it simply unregisters the ipcns callback
routine as soon as msgmni has been changed from procfs or sysctl().

Even if msgmni is set by hand, it should be possible to make it back
automatically recomputed upon memory add/remove or ipcns creation/removal.
This what is achieved in patch 8: if set to a negative value, msgmni is added
back to the ipcns notifier chain, making it automatically recomputed again.

These patches should be applied to 2.6.24-mm1, in the following order:

[PATCH 1/8]: ipc_scale_msgmni_with_lowmem.patch
[PATCH 2/8]: ipc_scale_msgmni_with_namespaces.patch
[PATCH 3/8]: ipc_slab_memory_callback_prio_to_const.patch
[PATCH 4/8]: ipc_recompute_msgmni_on_memory_hotplug.patch
[PATCH 5/8]: ipc_ipcns_notification_workqueue.patch
[PATCH 6/8]: ipc_recompute_msgmni_on_ipcns_create_remove.patch
[PATCH 7/8]: ipc_unregister_callback_on_msgmni_manual_change.patch
[PATCH 8/8]: ipc_register_callback_on_negative_msgmni.patch

Andrew, do you think this patchset could be considered for inclusion in -mm
(or at least patches 1 to 6)?

Regards,
Nadia

--

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

---

Subject: [PATCH 4/8] Recomputing msgmni on memory add / remove
Posted by Nadia Derbey on Mon, 11 Feb 2008 14:16:50 GMT

[PATCH 04/08]

This patch introduces the registration of a callback routine that recomputes
msg_ctlmni upon memory add / remove.

A single notifier block is registered in the hotplug memory chain for all the
ipc namespaces.

Since the ipc namespaces are not linked together, they have their own
notification chain: one notifier_block is defined per ipc namespace.

Each time an ipc namespace is created (removed) it registers (unregisters)
its notifier block in (from) the ipcns chain.
The callback routine registered in the memory chain invokes the ipcns
notifier chain with the IPCNS_LOWMEM event.
Each callback routine registered in the ipcns namespace, in turn, recomputes
msgmni for the owning namespace.

Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

---
 include/linux/ipc_namespace.h | 43 +++++++++++++++++++++-
 include/linux/memory.h | 1
 ipc/Makefile | 3 +
 ipc/ipcns_notifier.c | 71 ++
 ipc/msg.c | 2 -
 ipc/namespace.c | 11 ++++++
 ipc/util.c | 33 +++++++++++++++++
 ipc/util.h | 2 +
 8 files changed, 162 insertions(+), 4 deletions(-)

Index: linux-2.6.24-mm1/include/linux/ipc_namespace.h
===
--- linux-2.6.24-mm1.orig/include/linux/ipc_namespace.h 2008-02-07 15:26:53.000000000 +0100
+++ linux-2.6.24-mm1/include/linux/ipc_namespace.h 2008-02-08 08:29:21.000000000 +0100
@@ -4,6 +4,17 @@
 #include <linux/err.h>
 #include <linux/idr.h>
 #include <linux/rwsem.h>
+#ifdef CONFIG_MEMORY_HOTPLUG
+#include <linux/notifier.h>
+#endif /* CONFIG_MEMORY_HOTPLUG */
+
+/*
+ * ipc namespace events
+ */

```
+#define IPCNS_MEMCHANGED   0x00000001   /* Notify lowmem size changed */
+
+#define IPCNS_CALLBACK_PRI 0
+

 struct ipc_ids {
  int in_use;
@@ -30,6 +41,10 @@ struct ipc_namespace {
  size_t  shm_ctlall;
  int  shm_ctlmni;
  int  shm_tot;
+
+#ifdef CONFIG_MEMORY_HOTPLUG
+ struct notifier_block ipcns_nb;
+#endif
 };

 extern struct ipc_namespace init_ipc_ns;
@@ -37,9 +52,33 @@ extern atomic_t nr_ipc_ns;

 #ifdef CONFIG_SYSVIPC
 #define INIT_IPC_NS(ns)  .ns  = &init_ipc_ns,
-#else
+
+#ifdef CONFIG_MEMORY_HOTPLUG
+
+extern int register_ipcns_notifier(struct ipc_namespace *);
+extern int unregister_ipcns_notifier(struct ipc_namespace *);
+extern int ipcns_notify(unsigned long);
+
+#else /* CONFIG_MEMORY_HOTPLUG */
+
+static inline int register_ipcns_notifier(struct ipc_namespace *ipcns)
+{
+ return 0;
+}
+static inline int unregister_ipcns_notifier(struct ipc_namespace *ipcns)
+{
+ return 0;
+}
+static inline int ipcns_notify(unsigned long ev)
+{
+ return 0;
+}
+
+#endif /* CONFIG_MEMORY_HOTPLUG */
+
+#else /* CONFIG_SYSVIPC */
```

```
 #define INIT_IPC_NS(ns)
-#endif
+#endif /* CONFIG_SYSVIPC */

 #if defined(CONFIG_SYSVIPC) && defined(CONFIG_IPC_NS)
 extern void free_ipc_ns(struct kref *kref);
Index: linux-2.6.24-mm1/include/linux/memory.h
===================================================================
--- linux-2.6.24-mm1.orig/include/linux/memory.h 2008-02-07 17:10:07.000000000 +0100
+++ linux-2.6.24-mm1/include/linux/memory.h 2008-02-08 08:04:36.000000000 +0100
@@ -58,6 +58,7 @@ struct mem_section;
  * order in the callback chain)
  */
 #define SLAB_CALLBACK_PRI       1
+#define IPC_CALLBACK_PRI        10

 #ifndef CONFIG_MEMORY_HOTPLUG_SPARSE
 static inline int memory_dev_init(void)
Index: linux-2.6.24-mm1/ipc/ipcns_notifier.c
===================================================================
--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.24-mm1/ipc/ipcns_notifier.c 2008-02-08 13:59:26.000000000 +0100
@@ -0,0 +1,71 @@
+/*
+ * linux/ipc/ipcns_notifier.c
+ * Copyright (C) 2007 BULL SA. Nadia Derbey
+ *
+ * Notification mechanism for ipc namespaces:
+ * The callback routine registered in the memory chain invokes the ipcns
+ * notifier chain with the IPCNS_MEMCHANGED event.
+ * Each callback routine registered in the ipcns namespace recomputes msgmni
+ * for the owning namespace.
+ */
+
+#include <linux/msg.h>
+#include <linux/rcupdate.h>
+#include <linux/notifier.h>
+#include <linux/nsproxy.h>
+#include <linux/ipc_namespace.h>
+
+#include "util.h"
+
+
+
+static BLOCKING_NOTIFIER_HEAD(ipcns_chain);
+
+
+static int ipcns_callback(struct notifier_block *self,
```

```
+    unsigned long action, void *arg)
+{
+ struct ipc_namespace *ns;
+
+ switch (action) {
+ case IPCNS_MEMCHANGED:   /* amount of lowmem has changed */
+  /*
+   * It's time to recompute msgmni
+   */
+  ns = container_of(self, struct ipc_namespace, ipcns_nb);
+  /*
+   * No need to get a reference on the ns: the 1st job of
+   * free_ipc_ns() is to unregister the callback routine.
+   * blocking_notifier_chain_unregister takes the wr lock to do
+   * it.
+   * When this callback routine is called the rd lock is held by
+   * blocking_notifier_call_chain.
+   * So the ipc ns cannot be freed while we are here.
+   */
+  recompute_msgmni(ns);
+  break;
+ default:
+  break;
+ }
+
+ return NOTIFY_OK;
+}
+
+int register_ipcns_notifier(struct ipc_namespace *ns)
+{
+ memset(&ns->ipcns_nb, 0, sizeof(ns->ipcns_nb));
+ ns->ipcns_nb.notifier_call = ipcns_callback;
+ ns->ipcns_nb.priority = IPCNS_CALLBACK_PRI;
+ return blocking_notifier_chain_register(&ipcns_chain, &ns->ipcns_nb);
+}
+
+int unregister_ipcns_notifier(struct ipc_namespace *ns)
+{
+ return blocking_notifier_chain_unregister(&ipcns_chain,
+       &ns->ipcns_nb);
+}
+
+int ipcns_notify(unsigned long val)
+{
+ return blocking_notifier_call_chain(&ipcns_chain, val, NULL);
+}
Index: linux-2.6.24-mm1/ipc/Makefile
===================================================================
```

```
--- linux-2.6.24-mm1.orig/ipc/Makefile 2008-02-07 13:41:07.000000000 +0100
+++ linux-2.6.24-mm1/ipc/Makefile 2008-02-08 08:10:13.000000000 +0100
@@ -3,7 +3,8 @@
 #

 obj-$(CONFIG_SYSVIPC_COMPAT) += compat.o
-obj-$(CONFIG_SYSVIPC) += util.o msgutil.o msg.o sem.o shm.o
+obj_mem-$(CONFIG_MEMORY_HOTPLUG) += ipcns_notifier.o
+obj-$(CONFIG_SYSVIPC) += util.o msgutil.o msg.o sem.o shm.o $(obj_mem-y)
 obj-$(CONFIG_SYSVIPC_SYSCTL) += ipc_sysctl.o
 obj_mq-$(CONFIG_COMPAT) += compat_mq.o
 obj-$(CONFIG_POSIX_MQUEUE) += mqueue.o msgutil.o $(obj_mq-y)
Index: linux-2.6.24-mm1/ipc/util.c
===================================================================
--- linux-2.6.24-mm1.orig/ipc/util.c 2008-02-07 15:36:22.000000000 +0100
+++ linux-2.6.24-mm1/ipc/util.c 2008-02-08 08:15:35.000000000 +0100
@@ -33,6 +33,7 @@
 #include <linux/audit.h>
 #include <linux/nsproxy.h>
 #include <linux/rwsem.h>
+#include <linux/memory.h>
 #include <linux/ipc_namespace.h>

 #include <asm/unistd.h>
@@ -55,11 +56,41 @@ struct ipc_namespace init_ipc_ns = {
 atomic_t nr_ipc_ns = ATOMIC_INIT(1);


+#ifdef CONFIG_MEMORY_HOTPLUG
+
+static int ipc_memory_callback(struct notifier_block *self,
+    unsigned long action, void *arg)
+{
+ switch (action) {
+ case MEM_ONLINE:    /* memory successfully brought online */
+ case MEM_OFFLINE:   /* or offline: it's time to recompute msgmni */
+  /*
+   * This is done by invoking the ipcns notifier chain with the
+   * IPC_MEMCHANGED event.
+   */
+  ipcns_notify(IPCNS_MEMCHANGED);
+  break;
+ case MEM_GOING_ONLINE:
+ case MEM_GOING_OFFLINE:
+ case MEM_CANCEL_ONLINE:
+ case MEM_CANCEL_OFFLINE:
+ default:
+  break;
```

```
+ }
+
+ return NOTIFY_OK;
+}
+
+#endif /* CONFIG_MEMORY_HOTPLUG */
+
 /**
  * ipc_init - initialise IPC subsystem
  *
  * The various system5 IPC resources (semaphores, messages and shared
  * memory) are initialised
+ * A callback routine is registered into the memory hotplug notifier
+ * chain: since msgmni scales to lowmem this callback routine will be
+ * called upon successful memory add / remove to recompute msmgni.
  */

 static int __init ipc_init(void)
@@ -67,6 +98,8 @@ static int __init ipc_init(void)
  sem_init();
  msg_init();
  shm_init();
+ hotplug_memory_notifier(ipc_memory_callback, IPC_CALLBACK_PRI);
+ register_ipcns_notifier(&init_ipc_ns);
  return 0;
 }
 __initcall(ipc_init);
Index: linux-2.6.24-mm1/ipc/namespace.c
===================================================================
--- linux-2.6.24-mm1.orig/ipc/namespace.c 2008-02-07 15:40:19.000000000 +0100
+++ linux-2.6.24-mm1/ipc/namespace.c 2008-02-08 08:18:35.000000000 +0100
@@ -26,6 +26,8 @@ static struct ipc_namespace *clone_ipc_n
  msg_init_ns(ns);
  shm_init_ns(ns);

+ register_ipcns_notifier(ns);
+
  kref_init(&ns->kref);
  return ns;
 }
@@ -81,6 +83,15 @@ void free_ipc_ns(struct kref *kref)
  struct ipc_namespace *ns;

  ns = container_of(kref, struct ipc_namespace, kref);
+ /*
+  * Unregistering the hotplug notifier at the beginning guarantees
+  * that the ipc namespace won't be freed while we are inside the
+  * callback routine. Since the blocking_notifier_chain_XXX routines
```

```
+  * hold a rw lock on the notifier list, unregister_ipcns_notifier()
+  * won't take the rw lock before blocking_notifier_call_chain() has
+  * released the rd lock.
+  */
+ unregister_ipcns_notifier(ns);
  sem_exit_ns(ns);
  msg_exit_ns(ns);
  shm_exit_ns(ns);
Index: linux-2.6.24-mm1/ipc/msg.c
===================================================================
--- linux-2.6.24-mm1.orig/ipc/msg.c 2008-02-07 15:43:51.000000000 +0100
+++ linux-2.6.24-mm1/ipc/msg.c 2008-02-08 08:19:36.000000000 +0100
@@ -85,7 +85,7 @@ static int sysvipc_msg_proc_show(struct
  * Also take into account the number of nsproxies created so far.
  * This should be done staying within the (MSGMNI , IPCMNI/nr_ipc_ns) range.
  */
-static void recompute_msgmni(struct ipc_namespace *ns)
+void recompute_msgmni(struct ipc_namespace *ns)
 {
  struct sysinfo i;
  unsigned long allowed;
Index: linux-2.6.24-mm1/ipc/util.h
===================================================================
--- linux-2.6.24-mm1.orig/ipc/util.h 2008-02-07 13:41:07.000000000 +0100
+++ linux-2.6.24-mm1/ipc/util.h 2008-02-08 08:21:23.000000000 +0100
@@ -124,6 +124,8 @@ extern void free_msg(struct msg_msg *msg
 extern struct msg_msg *load_msg(const void __user *src, int len);
 extern int store_msg(void __user *dest, struct msg_msg *msg, int len);

+extern void recompute_msgmni(struct ipc_namespace *);
+
 static inline int ipc_buildid(int id, int seq)
 {
  return SEQ_MULTIPLIER * seq + id;

--

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

Subject: [PATCH 6/8] Recomputing msgmni on ipc namespace creation/removal
Posted by Nadia Derbey on Mon, 11 Feb 2008 14:16:52 GMT
View Forum Message <> Reply to Message

[PATCH 06/08]

This patch introduces a notification mechanism that aims at recomputing
msgmni each time an ipc namespace is created or removed.

The ipc namespace notifier chain already defined for memory hotplug management
is used for that purpose too.

Each time a new ipc namespace is allocated or an existing ipc namespace is
removed, the ipcns notifier chain is notified. The callback routine for each
registered ipc namespace is then activated in order to recompute msgmni for
that namespace.

Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

---
 include/linux/ipc_namespace.h | 25 ++----------------------
 ipc/Makefile | 3 +--
 ipc/ipcns_notifier.c | 2 ++
 ipc/namespace.c | 12 ++++++++++++
 4 files changed, 17 insertions(+), 25 deletions(-)

Index: linux-2.6.24-mm1/include/linux/ipc_namespace.h
===
--- linux-2.6.24-mm1.orig/include/linux/ipc_namespace.h 2008-02-08 08:29:21.000000000 +0100
+++ linux-2.6.24-mm1/include/linux/ipc_namespace.h 2008-02-08 14:35:08.000000000 +0100
@@ -4,14 +4,14 @@
 #include <linux/err.h>
 #include <linux/idr.h>
 #include <linux/rwsem.h>
-#ifdef CONFIG_MEMORY_HOTPLUG
 #include <linux/notifier.h>
-#endif /* CONFIG_MEMORY_HOTPLUG */

 /*
  * ipc namespace events
  */
 #define IPCNS_MEMCHANGED   0x00000001   /* Notify lowmem size changed */
+#define IPCNS_CREATED 0x00000002   /* Notify new ipc namespace created */
+#define IPCNS_REMOVED 0x00000003   /* Notify ipc namespace removed */

 #define IPCNS_CALLBACK_PRI 0

@@ -42,9 +42,7 @@ struct ipc_namespace {
  int shm_ctlmni;
  int shm_tot;

-#ifdef CONFIG_MEMORY_HOTPLUG
  struct notifier_block ipcns_nb;

```
-#endif
 };

 extern struct ipc_namespace init_ipc_ns;
@@ -53,29 +51,10 @@ extern atomic_t nr_ipc_ns;
 #ifdef CONFIG_SYSVIPC
 #define INIT_IPC_NS(ns)  .ns  = &init_ipc_ns,

-#ifdef CONFIG_MEMORY_HOTPLUG
-
 extern int register_ipcns_notifier(struct ipc_namespace *);
 extern int unregister_ipcns_notifier(struct ipc_namespace *);
 extern int ipcns_notify(unsigned long);

-#else /* CONFIG_MEMORY_HOTPLUG */
-
-static inline int register_ipcns_notifier(struct ipc_namespace *ipcns)
-{
- return 0;
-}
-static inline int unregister_ipcns_notifier(struct ipc_namespace *ipcns)
-{
- return 0;
-}
-static inline int ipcns_notify(unsigned long ev)
-{
- return 0;
-}
-
-#endif /* CONFIG_MEMORY_HOTPLUG */
-
 #else /* CONFIG_SYSVIPC */
 #define INIT_IPC_NS(ns)
 #endif /* CONFIG_SYSVIPC */
Index: linux-2.6.24-mm1/ipc/ipcns_notifier.c
===================================================================
--- linux-2.6.24-mm1.orig/ipc/ipcns_notifier.c 2008-02-08 13:59:26.000000000 +0100
+++ linux-2.6.24-mm1/ipc/ipcns_notifier.c 2008-02-08 14:36:05.000000000 +0100
@@ -29,6 +29,8 @@ static int ipcns_callback(struct notifie

  switch (action) {
  case IPCNS_MEMCHANGED:   /* amount of lowmem has changed */
+ case IPCNS_CREATED:
+ case IPCNS_REMOVED:
   /*
    * It's time to recompute msgmni
    */
Index: linux-2.6.24-mm1/ipc/Makefile
```

```
===================================================================
--- linux-2.6.24-mm1.orig/ipc/Makefile 2008-02-08 08:10:13.000000000 +0100
+++ linux-2.6.24-mm1/ipc/Makefile 2008-02-08 14:36:52.000000000 +0100
@@ -3,8 +3,7 @@
 #

 obj-$(CONFIG_SYSVIPC_COMPAT) += compat.o
-obj_mem-$(CONFIG_MEMORY_HOTPLUG) += ipcns_notifier.o
-obj-$(CONFIG_SYSVIPC) += util.o msgutil.o msg.o sem.o shm.o $(obj_mem-y)
+obj-$(CONFIG_SYSVIPC) += util.o msgutil.o msg.o sem.o shm.o ipcns_notifier.o
 obj-$(CONFIG_SYSVIPC_SYSCTL) += ipc_sysctl.o
 obj_mq-$(CONFIG_COMPAT) += compat_mq.o
 obj-$(CONFIG_POSIX_MQUEUE) += mqueue.o msgutil.o $(obj_mq-y)
Index: linux-2.6.24-mm1/ipc/namespace.c
===================================================================
--- linux-2.6.24-mm1.orig/ipc/namespace.c 2008-02-08 08:18:35.000000000 +0100
+++ linux-2.6.24-mm1/ipc/namespace.c 2008-02-08 14:41:37.000000000 +0100
@@ -26,6 +26,12 @@ static struct ipc_namespace *clone_ipc_n
  msg_init_ns(ns);
  shm_init_ns(ns);

+ /*
+  * msgmni has already been computed for the new ipc ns.
+  * Thus, do the ipcns creation notification before registering that
+  * new ipcns in the chain.
+  */
+ ipcns_notify(IPCNS_CREATED);
  register_ipcns_notifier(ns);

  kref_init(&ns->kref);
@@ -97,4 +103,10 @@ void free_ipc_ns(struct kref *kref)
  shm_exit_ns(ns);
  kfree(ns);
  atomic_dec(&nr_ipc_ns);
+
+ /*
+  * Do the ipcns removal notification after decrementing nr_ipc_ns in
+  * order to have a correct value when recomputing msgmni.
+  */
+ ipcns_notify(IPCNS_REMOVED);
 }

--
```

Subject: [PATCH 7/8] Do not recompute msgmni anymore if explicitely set by user
Posted by Nadia Derbey on Mon, 11 Feb 2008 14:16:53 GMT
View Forum Message <> Reply to Message

[PATCH 07/08]

This patch makes msgmni not recomputed anymore upon ipc namespace creation / removal or memory add/remove, as soon as it has been set from userland.

As soon as msgmni is explicitely set via procfs or sysctl(), the associated callback routine is unregistered from the ipc namespace notifier chain.

Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

```
---
 ipc/ipc_sysctl.c |   43 +++++++++++++++++++++++++++++++++++++++++--
 1 file changed, 41 insertions(+), 2 deletions(-)

Index: linux-2.6.24-mm1/ipc/ipc_sysctl.c
===================================================================
--- linux-2.6.24-mm1.orig/ipc/ipc_sysctl.c 2008-02-08 16:07:15.000000000 +0100
+++ linux-2.6.24-mm1/ipc/ipc_sysctl.c 2008-02-08 16:08:32.000000000 +0100
@@ -35,6 +35,24 @@ static int proc_ipc_dointvec(ctl_table *
  return proc_dointvec(&ipc_table, write, filp, buffer, lenp, ppos);
 }

+static int proc_ipc_callback_dointvec(ctl_table *table, int write,
+ struct file *filp, void __user *buffer, size_t *lenp, loff_t *ppos)
+{
+ size_t lenp_bef = *lenp;
+ int rc;
+
+ rc = proc_ipc_dointvec(table, write, filp, buffer, lenp, ppos);
+
+ if (write && !rc && lenp_bef == *lenp)
+  /*
+   * Tunable has successfully been changed from userland:
+   * disable its automatic recomputing.
+   */
+  unregister_ipcns_notifier(current->nsproxy->ipc_ns);
+
+ return rc;
+}
+
 static int proc_ipc_doulongvec_minmax(ctl_table *table, int write,
  struct file *filp, void __user *buffer, size_t *lenp, loff_t *ppos)
 {
@@ -49,6 +67,7 @@ static int proc_ipc_doulongvec_minmax(ct
```

```
 #else
 #define proc_ipc_doulongvec_minmax NULL
 #define proc_ipc_dointvec    NULL
+#define proc_ipc_callback_dointvec NULL
 #endif

 #ifdef CONFIG_SYSCTL_SYSCALL
@@ -90,8 +109,28 @@ static int sysctl_ipc_data(ctl_table *ta
  }
  return 1;
 }
+
+static int sysctl_ipc_registered_data(ctl_table *table, int __user *name,
+  int nlen, void __user *oldval, size_t __user *oldlenp,
+  void __user *newval, size_t newlen)
+{
+ int rc;
+
+ rc = sysctl_ipc_data(table, name, nlen, oldval, oldlenp, newval,
+  newlen);
+
+ if (newval && newlen && rc > 0)
+  /*
+   * Tunable has successfully been changed from userland:
+   * disable its automatic recomputing.
+   */
+  unregister_ipcns_notifier(current->nsproxy->ipc_ns);
+
+ return rc;
+}
 #else
 #define sysctl_ipc_data NULL
+#define sysctl_ipc_registered_data NULL
 #endif

 static struct ctl_table ipc_kern_table[] = {
@@ -137,8 +176,8 @@ static struct ctl_table ipc_kern_table[]
   .data  = &init_ipc_ns.msg_ctlmni,
   .maxlen  = sizeof (init_ipc_ns.msg_ctlmni),
   .mode  = 0644,
-  .proc_handler = proc_ipc_dointvec,
-  .strategy = sysctl_ipc_data,
+  .proc_handler = proc_ipc_callback_dointvec,
+  .strategy = sysctl_ipc_registered_data,
  },
  {
   .ctl_name = KERN_MSGMNB,
```

--

_____

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

---

Subject: [PATCH 8/8] Re-enable msgmni automatic recomputing msgmni if set to negative
Posted by Nadia Derbey on Mon, 11 Feb 2008 14:16:54 GMT
View Forum Message <> Reply to Message

[PATCH 08/08]

This patch is the enhancement as asked for by Yasunori: if msgmni is set to
a negative value, register it back into the ipcns notifier chain.

A new interface has been added to the notification mechanism:
notifier_chain_cond_register() registers a notifier block only if not already
registered. With that new interface we avoid taking care of the states changes
in procfs.

Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

---
 include/linux/ipc_namespace.h | 1
 include/linux/notifier.h | 4 +++
 ipc/ipc_sysctl.c | 45 ++++++++++++++++++++++++++++++----------
 ipc/ipcns_notifier.c | 9 ++++++++
 kernel/notifier.c | 38 ++++++++++++++++++++++++++++++++++++
 5 files changed, 87 insertions(+), 10 deletions(-)

Index: linux-2.6.24-mm1/ipc/ipc_sysctl.c
===
--- linux-2.6.24-mm1.orig/ipc/ipc_sysctl.c 2008-02-08 16:08:32.000000000 +0100
+++ linux-2.6.24-mm1/ipc/ipc_sysctl.c 2008-02-11 14:32:09.000000000 +0100
@@ -15,6 +15,8 @@
 #include <linux/sysctl.h>
 #include <linux/uaccess.h>
 #include <linux/ipc_namespace.h>
+#include <linux/msg.h>
+#include "util.h"

 static void *get_ipc(ctl_table *table)
 {
@@ -24,6 +26,27 @@ static void *get_ipc(ctl_table *table)
  return which;
 }

```
+/*
+ * Routine that is called when a tunable has successfully been changed by
+ * hand and it has a callback routine registered on the ipc namespace notifier
+ * chain: we don't want such tunables to be recomputed anymore upon memory
+ * add/remove or ipc namespace creation/removal.
+ * They can come back to a recomputable state by being set to a <0 value.
+ */
+static void tunable_set_callback(int val)
+{
+ if (val >= 0)
+  unregister_ipcns_notifier(current->nsproxy->ipc_ns);
+ else {
+  /*
+   * Re-enable automatic recomputing only if not already
+   * enabled.
+   */
+  recompute_msgmni(current->nsproxy->ipc_ns);
+  cond_register_ipcns_notifier(current->nsproxy->ipc_ns);
+ }
+}
+
 #ifdef CONFIG_PROC_FS
 static int proc_ipc_dointvec(ctl_table *table, int write, struct file *filp,
  void __user *buffer, size_t *lenp, loff_t *ppos)
@@ -38,17 +61,17 @@ static int proc_ipc_dointvec(ctl_table *
 static int proc_ipc_callback_dointvec(ctl_table *table, int write,
  struct file *filp, void __user *buffer, size_t *lenp, loff_t *ppos)
 {
+ struct ctl_table ipc_table;
  size_t lenp_bef = *lenp;
  int rc;

- rc = proc_ipc_dointvec(table, write, filp, buffer, lenp, ppos);
+ memcpy(&ipc_table, table, sizeof(ipc_table));
+ ipc_table.data = get_ipc(table);
+
+ rc = proc_dointvec(&ipc_table, write, filp, buffer, lenp, ppos);

  if (write && !rc && lenp_bef == *lenp)
-  /*
-   * Tunable has successfully been changed from userland:
-   * disable its automatic recomputing.
-   */
-  unregister_ipcns_notifier(current->nsproxy->ipc_ns);
+  tunable_set_callback(*((int *)(ipc_table.data)));

  return rc;
```

```
 }
@@ -119,12 +142,14 @@ static int sysctl_ipc_registered_data(ct
  rc = sysctl_ipc_data(table, name, nlen, oldval, oldlenp, newval,
   newlen);

- if (newval && newlen && rc > 0)
+ if (newval && newlen && rc > 0) {
   /*
-   * Tunable has successfully been changed from userland:
-   * disable its automatic recomputing.
+   * Tunable has successfully been changed from userland
    */
-  unregister_ipcns_notifier(current->nsproxy->ipc_ns);
+  int *data = get_ipc(table);
+
+  tunable_set_callback(*data);
+ }

  return rc;
 }
Index: linux-2.6.24-mm1/kernel/notifier.c
===================================================================
--- linux-2.6.24-mm1.orig/kernel/notifier.c 2008-02-07 13:40:46.000000000 +0100
+++ linux-2.6.24-mm1/kernel/notifier.c 2008-02-11 14:54:34.000000000 +0100
@@ -31,6 +31,21 @@ static int notifier_chain_register(struc
  return 0;
 }

+static int notifier_chain_cond_register(struct notifier_block **nl,
+  struct notifier_block *n)
+{
+ while ((*nl) != NULL) {
+  if ((*nl) == n)
+   return 0;
+  if (n->priority > (*nl)->priority)
+   break;
+  nl = &((*nl)->next);
+ }
+ n->next = *nl;
+ rcu_assign_pointer(*nl, n);
+ return 0;
+}
+
 static int notifier_chain_unregister(struct notifier_block **nl,
   struct notifier_block *n)
 {
@@ -205,6 +220,29 @@ int blocking_notifier_chain_register(str
 EXPORT_SYMBOL_GPL(blocking_notifier_chain_register);
```

```
 /**
+ * blocking_notifier_chain_cond_register - Cond add notifier to a blocking notifier chain
+ * @nh: Pointer to head of the blocking notifier chain
+ * @n: New entry in notifier chain
+ *
+ * Adds a notifier to a blocking notifier chain, only if not already
+ * present in the chain.
+ * Must be called in process context.
+ *
+ * Currently always returns zero.
+ */
+int blocking_notifier_chain_cond_register(struct blocking_notifier_head *nh,
+  struct notifier_block *n)
+{
+ int ret;
+
+ down_write(&nh->rwsem);
+ ret = notifier_chain_cond_register(&nh->head, n);
+ up_write(&nh->rwsem);
+ return ret;
+}
+EXPORT_SYMBOL_GPL(blocking_notifier_chain_cond_register);
+
+/**
  * blocking_notifier_chain_unregister - Remove notifier from a blocking notifier chain
  * @nh: Pointer to head of the blocking notifier chain
  * @n: Entry to remove from notifier chain
Index: linux-2.6.24-mm1/include/linux/notifier.h
===================================================================
--- linux-2.6.24-mm1.orig/include/linux/notifier.h 2008-02-07 13:40:35.000000000 +0100
+++ linux-2.6.24-mm1/include/linux/notifier.h 2008-02-11 14:55:06.000000000 +0100
@@ -121,6 +121,10 @@ extern int raw_notifier_chain_register(s
 extern int srcu_notifier_chain_register(struct srcu_notifier_head *nh,
   struct notifier_block *nb);

+extern int blocking_notifier_chain_cond_register(
+  struct blocking_notifier_head *nh,
+  struct notifier_block *nb);
+
 extern int atomic_notifier_chain_unregister(struct atomic_notifier_head *nh,
   struct notifier_block *nb);
 extern int blocking_notifier_chain_unregister(struct blocking_notifier_head *nh,
Index: linux-2.6.24-mm1/include/linux/ipc_namespace.h
===================================================================
--- linux-2.6.24-mm1.orig/include/linux/ipc_namespace.h 2008-02-08 14:35:08.000000000 +0100
+++ linux-2.6.24-mm1/include/linux/ipc_namespace.h 2008-02-11 11:19:31.000000000 +0100
@@ -52,6 +52,7 @@ extern atomic_t nr_ipc_ns;
```

```
 #define INIT_IPC_NS(ns)  .ns  = &init_ipc_ns,

 extern int register_ipcns_notifier(struct ipc_namespace *);
+extern int cond_register_ipcns_notifier(struct ipc_namespace *);
 extern int unregister_ipcns_notifier(struct ipc_namespace *);
 extern int ipcns_notify(unsigned long);

Index: linux-2.6.24-mm1/ipc/ipcns_notifier.c
===================================================================
--- linux-2.6.24-mm1.orig/ipc/ipcns_notifier.c 2008-02-08 14:36:05.000000000 +0100
+++ linux-2.6.24-mm1/ipc/ipcns_notifier.c 2008-02-11 12:24:39.000000000 +0100
@@ -61,6 +61,15 @@ int register_ipcns_notifier(struct ipc_n
  return blocking_notifier_chain_register(&ipcns_chain, &ns->ipcns_nb);
 }

+int cond_register_ipcns_notifier(struct ipc_namespace *ns)
+{
+ memset(&ns->ipcns_nb, 0, sizeof(ns->ipcns_nb));
+ ns->ipcns_nb.notifier_call = ipcns_callback;
+ ns->ipcns_nb.priority = IPCNS_CALLBACK_PRI;
+ return blocking_notifier_chain_cond_register(&ipcns_chain,
+       &ns->ipcns_nb);
+}
+
 int unregister_ipcns_notifier(struct ipc_namespace *ns)
 {
  return blocking_notifier_chain_unregister(&ipcns_chain,

--
```

_____

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

---

Subject: Re: [PATCH 7/8] Do not recompute msgmni anymore if explicitely set by user

Posted by akpm on Mon, 11 Feb 2008 20:24:08 GMT
View Forum Message <> Reply to Message

On Mon, 11 Feb 2008 15:16:53 +0100
Nadia.Derbey@bull.net wrote:

> [PATCH 07/08]
>
> This patch makes msgmni not recomputed anymore upon ipc namespace creation /
> removal or memory add/remove, as soon as it has been set from userland.
>

> As soon as msgmni is explicitely set via procfs or sysctl(), the associated
> callback routine is unregistered from the ipc namespace notifier chain.
>

The patch series looks pretty good.

```
> ===================================================================
> --- linux-2.6.24-mm1.orig/ipc/ipc_sysctl.c 2008-02-08 16:07:15.000000000 +0100
> +++ linux-2.6.24-mm1/ipc/ipc_sysctl.c 2008-02-08 16:08:32.000000000 +0100
> @@ -35,6 +35,24 @@ static int proc_ipc_dointvec(ctl_table *
>   return proc_dointvec(&ipc_table, write, filp, buffer, lenp, ppos);
>  }
>
> +static int proc_ipc_callback_dointvec(ctl_table *table, int write,
> + struct file *filp, void __user *buffer, size_t *lenp, loff_t *ppos)
> +{
> + size_t lenp_bef = *lenp;
> + int rc;
> +
> + rc = proc_ipc_dointvec(table, write, filp, buffer, lenp, ppos);
> +
> + if (write && !rc && lenp_bef == *lenp)
> +  /*
> +   * Tunable has successfully been changed from userland:
> +   * disable its automatic recomputing.
> +   */
> +  unregister_ipcns_notifier(current->nsproxy->ipc_ns);
> +
> + return rc;
> +}
```

If you haven't done so, could you please check that it all builds cleanly
with CONFIG_PROCFS=n, and that all code which isn't needed if procfs is
disabled is not present in the final binary?

Subject: Re: [PATCH 8/8] Re-enable msgmni automatic recomputing msgmni if set to negative
Posted by akpm on Mon, 11 Feb 2008 20:27:48 GMT
View Forum Message <> Reply to Message

On Mon, 11 Feb 2008 15:16:54 +0100

Nadia.Derbey@bull.net wrote:

> [PATCH 08/08]
>
> This patch is the enhancement as asked for by Yasunori: if msgmni is set to
> a negative value, register it back into the ipcns notifier chain.
>
> A new interface has been added to the notification mechanism:
> notifier_chain_cond_register() registers a notifier block only if not already
> registered. With that new interface we avoid taking care of the states changes
> in procfs.
>
> ...
>
> static int proc_ipc_callback_dointvec(ctl_table *table, int write,
>   struct file *filp, void __user *buffer, size_t *lenp, loff_t *ppos)
> {
> + struct ctl_table ipc_table;
>   size_t lenp_bef = *lenp;
>   int rc;
>
> - rc = proc_ipc_dointvec(table, write, filp, buffer, lenp, ppos);
> + memcpy(&ipc_table, table, sizeof(ipc_table));
> + ipc_table.data = get_ipc(table);
> +
> + rc = proc_dointvec(&ipc_table, write, filp, buffer, lenp, ppos);
>
>   if (write && !rc && lenp_bef == *lenp)
> - /*
> -   * Tunable has successfully been changed from userland:
> -   * disable its automatic recomputing.
> -   */
> - unregister_ipcns_notifier(current->nsproxy->ipc_ns);
> + tunable_set_callback(*((int *)(ipc_table.data)));
>
>   return rc;
> }
> @@ -119,12 +142,14 @@ static int sysctl_ipc_registered_data(ct
>   rc = sysctl_ipc_data(table, name, nlen, oldval, oldlenp, newval,
>   newlen);
>
> - if (newval && newlen && rc > 0)
> + if (newval && newlen && rc > 0) {
>   /*
> -   * Tunable has successfully been changed from userland:
> -   * disable its automatic recomputing.
> +   * Tunable has successfully been changed from userland
>   */

> - unregister_ipcns_notifier(current->nsproxy->ipc_ns);
> + int *data = get_ipc(table);
> +
> + tunable_set_callback(*data);
> + }
>
> return rc;
> }

hm, what's happening here? We take a local copy of the caller's ctl_table
and then pass that into proc_dointvec(). Is that as hacky as it seems??

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

---

Subject: Re: [PATCH 7/8] Do not recompute msgmni anymore if explicitely set by user
Posted by Nadia Derbey on Tue, 12 Feb 2008 09:32:31 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Mon, 11 Feb 2008 15:16:53 +0100
> Nadia.Derbey@bull.net wrote:
>
>
>>[PATCH 07/08]
>>
>>This patch makes msgmni not recomputed anymore upon ipc namespace creation /
>>removal or memory add/remove, as soon as it has been set from userland.
>>
>>As soon as msgmni is explicitely set via procfs or sysctl(), the associated
>>callback routine is unregistered from the ipc namespace notifier chain.
>>
>
>
> The patch series looks pretty good.
>
>
>>==========
>>--- linux-2.6.24-mm1.orig/ipc/ipc_sysctl.c 2008-02-08 16:07:15.000000000 +0100
>>+++ linux-2.6.24-mm1/ipc/ipc_sysctl.c 2008-02-08 16:08:32.000000000 +0100
>>@@ -35,6 +35,24 @@ static int proc_ipc_dointvec(ctl_table *
>> return proc_dointvec(&ipc_table, write, filp, buffer, lenp, ppos);
>> }

>>
>>+static int proc_ipc_callback_dointvec(ctl_table *table, int write,
>>+ struct file *filp, void __user *buffer, size_t *lenp, loff_t *ppos)
>>+{
>>+ size_t lenp_bef = *lenp;
>>+ int rc;
>>+
>>+ rc = proc_ipc_dointvec(table, write, filp, buffer, lenp, ppos);
>>+
>>+ if (write && !rc && lenp_bef == *lenp)
>>+ /*
>>+   * Tunable has successfully been changed from userland:
>>+   * disable its automatic recomputing.
>>+   */
>>+ unregister_ipcns_notifier(current->nsproxy->ipc_ns);
>>+
>>+ return rc;
>>+}
>
>
> If you haven't done so, could you please check that it all builds cleanly
> with CONFIG_PROCFS=n, and that all code which isn't needed if procfs is
> disabled is not present in the final binary?
>
>
>
>

Andrew,

it builds fine, modulo some changes in ipv4 and ipv6 (see attached patch
- didn't find it in the hot fixes).

Regards,
Nadia

Subject: Re: [PATCH 7/8] Do not recompute msgmni anymore if explicitely set by user
Posted by akpm on Tue, 12 Feb 2008 09:44:44 GMT
View Forum Message <> Reply to Message

On Tue, 12 Feb 2008 10:32:31 +0100 Nadia Derbey <Nadia.Derbey@bull.net> wrote:

> it builds fine, modulo some changes in ipv4 and ipv6 (see attached patch
> - didn't find it in the hot fixes).

OK, thanks for checking. Did you confirm that we don't have unneeded code
in vmlinux when CONFIG_PROCFS=n? I guess before-and-after comparison of
the size(1) output would tell us.

Those networking build problems appear to have already been fixed.

In future, please quote the compiler error output in the changelog when
sending build fixes or warning fixes, thanks.

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

---

Subject: Re: [PATCH 7/8] Do not recompute msgmni anymore if explicitely set by user
Posted by Nadia Derbey on Tue, 12 Feb 2008 09:45:27 GMT
View Forum Message <> Reply to Message

Nadia Derbey wrote:
> Andrew Morton wrote:
>
>> On Mon, 11 Feb 2008 15:16:53 +0100
>> Nadia.Derbey@bull.net wrote:
>>
>>
>>> [PATCH 07/08]
>>>
>>> This patch makes msgmni not recomputed anymore upon ipc namespace
>>> creation /
>>> removal or memory add/remove, as soon as it has been set from userland.
>>>
>>> As soon as msgmni is explicitely set via procfs or sysctl(), the
>>> associated
>>> callback routine is unregistered from the ipc namespace notifier chain.
>>>
>>
>>
>> The patch series looks pretty good.
>>
>>
>>> ==========

```
>>> --- linux-2.6.24-mm1.orig/ipc/ipc_sysctl.c    2008-02-08
>>> 16:07:15.000000000 +0100
>>> +++ linux-2.6.24-mm1/ipc/ipc_sysctl.c    2008-02-08
>>> 16:08:32.000000000 +0100
>>> @@ -35,6 +35,24 @@ static int proc_ipc_dointvec(ctl_table *
>>>     return proc_dointvec(&ipc_table, write, filp, buffer, lenp, ppos);
>>> }
>>>
>>> +static int proc_ipc_callback_dointvec(ctl_table *table, int write,
>>> +    struct file *filp, void __user *buffer, size_t *lenp, loff_t *ppos)
>>> +{
>>> +    size_t lenp_bef = *lenp;
>>> +    int rc;
>>> +
>>> +    rc = proc_ipc_dointvec(table, write, filp, buffer, lenp, ppos);
>>> +
>>> +    if (write && !rc && lenp_bef == *lenp)
>>> +        /*
>>> +         * Tunable has successfully been changed from userland:
>>> +         * disable its automatic recomputing.
>>> +         */
>>> +        unregister_ipcns_notifier(current->nsproxy->ipc_ns);
>>> +
>>> +    return rc;
>>> +}
>>
>>
>>
>> If you haven't done so, could you please check that it all builds cleanly
>> with CONFIG_PROCFS=n, and that all code which isn't needed if procfs is
>> disabled is not present in the final binary?
>>
>>
>>
>>
>
> Andrew,
>
> it builds fine, modulo some changes in ipv4 and ipv6 (see attached patch
> - didn't find it in the hot fixes).
>
> Regards,
> Nadia
>
>
```

Oops, forgot the patch. Thx Benjamin!

Subject: Re: [PATCH 8/8] Re-enable msgmni automatic recomputing msgmni if set to negative
Posted by Nadia Derbey on Tue, 12 Feb 2008 11:38:38 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Mon, 11 Feb 2008 15:16:54 +0100
> Nadia.Derbey@bull.net wrote:
>
>
>>[PATCH 08/08]
>>
>>This patch is the enhancement as asked for by Yasunori: if msgmni is set to
>>a negative value, register it back into the ipcns notifier chain.
>>
>>A new interface has been added to the notification mechanism:
>>notifier_chain_cond_register() registers a notifier block only if not already
>>registered. With that new interface we avoid taking care of the states changes
>>in procfs.
>>
>>...
>>
>> static int proc_ipc_callback_dointvec(ctl_table *table, int write,
>> struct file *filp, void __user *buffer, size_t *lenp, loff_t *ppos)
>> {
>>+ struct ctl_table ipc_table;
>> size_t lenp_bef = *lenp;
>> int rc;
>>
>>- rc = proc_ipc_dointvec(table, write, filp, buffer, lenp, ppos);
>>+ memcpy(&ipc_table, table, sizeof(ipc_table));
>>+ ipc_table.data = get_ipc(table);
>>+
>>+ rc = proc_dointvec(&ipc_table, write, filp, buffer, lenp, ppos);
>>
>> if (write && !rc && lenp_bef == *lenp)
>>- /*
>>-   * Tunable has successfully been changed from userland:
>>-   * disable its automatic recomputing.

>>-   */
>>- unregister_ipcns_notifier(current->nsproxy->ipc_ns);
>>+ tunable_set_callback(*((int *)(ipc_table.data)));
>>
>> return rc;
>> }
>>@@ -119,12 +142,14 @@ static int sysctl_ipc_registered_data(ct
>> rc = sysctl_ipc_data(table, name, nlen, oldval, oldlenp, newval,
>>   newlen);
>>
>>- if (newval && newlen && rc > 0)
>>+ if (newval && newlen && rc > 0) {
>>   /*
>>-   * Tunable has successfully been changed from userland:
>>-   * disable its automatic recomputing.
>>+   * Tunable has successfully been changed from userland
>>   */
>>- unregister_ipcns_notifier(current->nsproxy->ipc_ns);
>>+ int *data = get_ipc(table);
>>+
>>+ tunable_set_callback(*data);
>>+ }
>>
>> return rc;
>> }
>
>
> hm, what's happening here? We take a local copy of the caller's ctl_table
> and then pass that into proc_dointvec(). Is that as hacky as it seems??
>
>

Well, the caller's ctl_table contains the tunables addresses for the
init namspeace in its .data fields. While what needs to be passed in to
proc_dointvec() is the tunable address in the caller's namespace.
Since all the fields in ipc_kern_table[] are ok but the .data one, imho
it's correct to store it in a local copy and change the data field to
the appropirate one, before passing it to proc_dointvec().

Regards,
Nadia

Subject: Re: [PATCH 7/8] Do not recompute msgmni anymore if explicitely set by user
Posted by Nadia Derbey on Tue, 12 Feb 2008 15:15:00 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Tue, 12 Feb 2008 10:32:31 +0100 Nadia Derbey <Nadia.Derbey@bull.net> wrote:
>
>
>>it builds fine, modulo some changes in ipv4 and ipv6 (see attached patch
>>- didn't find it in the hot fixes).
>
>
> OK, thanks for checking. Did you confirm that we don't have unneeded code
> in vmlinux when CONFIG_PROCFS=n? I guess before-and-after comparison of
> the size(1) output would tell us.
>
> Those networking build problems appear to have already been fixed.
>
> In future, please quote the compiler error output in the changelog when
> sending build fixes or warning fixes, thanks.
>
>
>

BEFORE:

lkernel@akt$ size vmlinux
   text   data   bss   dec   hex filename
4318525 454484 462848 5235857 4fe491 vmlinux

AFTER:

lkernel@akt$ size vmlinux
   text   data   bss   dec   hex filename
4323161 454484 462848 5240493 4ff6ad vmlinux

which makes it +4636 = +0.11%

I've got the details for */built-in.o if needed.

Regards,
Nadia

Subject: Re: [PATCH 7/8] Do not recompute msgmni anymore if explicitely set by user
Posted by akpm on Tue, 12 Feb 2008 19:44:39 GMT
View Forum Message <> Reply to Message

On Tue, 12 Feb 2008 16:15:00 +0100
Nadia Derbey <Nadia.Derbey@bull.net> wrote:

> Andrew Morton wrote:
> > On Tue, 12 Feb 2008 10:32:31 +0100 Nadia Derbey <Nadia.Derbey@bull.net> wrote:
> >
> >
> > >>it builds fine, modulo some changes in ipv4 and ipv6 (see attached patch
> > >>- didn't find it in the hot fixes).
> >
> >
> > OK, thanks for checking. Did you confirm that we don't have unneeded code
> > in vmlinux when CONFIG_PROCFS=n? I guess before-and-after comparison of
> > the size(1) output would tell us.
> >
> > Those networking build problems appear to have already been fixed.
> >
> > In future, please quote the compiler error output in the changelog when
> > sending build fixes or warning fixes, thanks.
> >
> >
> >
> >
> BEFORE:
>
> lkernel@akt$ size vmlinux
>   text   data   bss   dec   hex filename
> 4318525 454484 462848 5235857 4fe491 vmlinux
>
>
> AFTER:
>
> lkernel@akt$ size vmlinux
>   text   data   bss   dec   hex filename
> 4323161 454484 462848 5240493 4ff6ad vmlinux
>
> which makes it +4636 = +0.11%
>
> I've got the details for */built-in.o if needed.
>

That seems to be a lot of increase. Are you sure you had CONFIG_PROCFS=n
in both cases? If so, the patch must have added a lot of code which will
never be executed?

Subject: Re: [PATCH 7/8] Do not recompute msgmni anymore if explicitely set by user
Posted by Nadia Derbey on Thu, 14 Feb 2008 11:47:57 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Tue, 12 Feb 2008 16:15:00 +0100
> Nadia Derbey <Nadia.Derbey@bull.net> wrote:
>
>
>>Andrew Morton wrote:
>>
>>>On Tue, 12 Feb 2008 10:32:31 +0100 Nadia Derbey <Nadia.Derbey@bull.net> wrote:
>>>
>>>
>>>
>>>>it builds fine, modulo some changes in ipv4 and ipv6 (see attached patch
>>>>- didn't find it in the hot fixes).
>>>
>>>
>>>OK, thanks for checking. Did you confirm that we don't have unneeded code
>>>in vmlinux when CONFIG_PROCFS=n? I guess before-and-after comparison of
>>>the size(1) output would tell us.
>>>
>>>Those networking build problems appear to have already been fixed.
>>>
>>>In future, please quote the compiler error output in the changelog when
>>>sending build fixes or warning fixes, thanks.
>>>
>>>
>>>
>>
>>BEFORE:
>>
>>lkernel@akt$ size vmlinux
>>   text   data   bss   dec   hex filename
>>4318525 454484 462848 5235857 4fe491 vmlinux
>>
>>
>>AFTER:
>>

>>lkernel@akt$ size vmlinux
>>   text   data   bss   dec   hex filename
>>4323161 454484 462848 5240493 4ff6ad vmlinux
>>
>>which makes it +4636 = +0.11%
>>
>>I've got the details for */built-in.o if needed.
>>
>
>
> That seems to be a lot of increase. Are you sure you had CONFIG_PROCFS=n
> in both cases? If so, the patch must have added a lot of code which will
> never be executed?
>
>
>
Well, the patches that are impacted by procfs being configured or not
are #7 and #8. While the sizes I've sent you are before all patches
applied vs after all patches applied :-(

So here are the "interesting sizes" - all with CONFIG_PROC_FS unset:

before patch 7:
   text   data   bss   dec   hex filename
4318757 454484 462848 5236089 4fe579 vmlinux

before patch 8:
   text   data   bss   dec   hex filename
4318853 454484 462848 5236185 4fe5d9 vmlinux   +96

after patch 8:
4319055 454484 462848 5236387 4fe6a3 vmlinux   +202

The higher difference after patch 8 is because I'm adding the new
interface blocking_notifier_chain_cond_register() even if CONFIG_PROC_FS
is not defined. This is to cover the case where msgmni is set through
the sysctl() syscall (CONFIG_SYSCTL_SYSCALL).

Regards,
Nadia