Subject: [-mm PATCH 0/9] Memory controller introduction (v5)
Posted by Balbir Singh on Mon, 13 Aug 2007 17:41:13 GMT
View Forum Message <> Reply to Message

Hi, Andrew,

Here's version 5 of the memory controller (against 2.6.23-rc1-mm1).

I've tested it and made several changes based on review comments from
several people in the community. I would consider this version as
ready for inclusion and thus request you to include it into the -mm tree.
Including it in the -mm tree would help

1. Iron out any major bugs
2. Get more review comments and testing in the community
3. Help it evolve iteratively

I do however that this version is *not* bug free, I am however committed
to fixing/resolving any issues reported against the patches/code.

Changelog since version 4

1. Renamed meta_page to page_container (Nick Piggin)
2. Moved locking from page flags to last bit of the page_container pointer
   (Nick Piggin)
3. Fixed a rare race in mem_container_isolate_pages (YAMAMOTO Takashi)

Changelog since version 3

1. Ported to v11 of the containers patchset (2.6.23-rc1-mm1). Paul Menage
   helped immensely with a detailed review of v3
2. Reclaim is retried to allow reclaim of pages coming in as a result
   of mapped pages reclaim (swap cache growing as a result of RSS reclaim)
3. page_referenced() is now container aware. During container reclaim,
   references from other containers do not prevent a page from being
   reclaimed from a non-referencing container
4. Fixed a possible race condition spotted by YAMAMOTO Takashi

Changelog since version 2

1. Improved error handling in mm/memory.c (spotted by YAMAMOTO Takashi)
2. Test results included
3. try_to_free_mem_container_pages() bug fix (sc->may_writepage is now
   set to !laptop_mode)

Changelog since version 1

1. Fixed some compile time errors (in mm/migrate.c from Vaidyanathan S)

2. Fixed a panic seen when LIST_DEBUG is enabled
3. Added a mechanism to control whether we track page cache or both
   page cache and mapped pages (as requested by Pavel)
4. Dave Hansen provided detail review comments on the code.

This patchset implements another version of the memory controller. These
patches have been through a big churn, the first set of patches were posted
last year and earlier this year at
 http://lkml.org/lkml/2007/2/19/10

This patchset draws from the patches listed above and from some of the
contents of the patches posted by Vaidyanathan for page cache control.
 http://lkml.org/lkml/2007/6/20/92

At OLS, the resource management BOF, it was discussed that we need to manage
RSS and unmapped page cache together. This patchset is a step towards that

TODO's

1. Add memory controller water mark support. Reclaim on high water mark
2. Add support for shrinking on limit change
3. Add per zone per container LRU lists (this is being actively worked
   on by Pavel Emelianov)
4. Figure out a better CLUI for the controller
5. Add better statistics
6. Explore using read_unit64() as recommended by Paul Menage
   (NOTE: read_ulong() would also be nice to have)

In case you have been using/testing the RSS controller, you'll find that
this controller works slower than the RSS controller. The reason being
that both swap cache and page cache is accounted for, so pages do go
out to swap upon reclaim (they cannot live in the swap cache).

Any test output, feedback, comments, suggestions are welcome! I am committed
to fixing any bugs and improving the performance of the memory controller.
Do not hesitate to send any fixes, request for fixes that is required.

Using the patches

1. Enable Memory controller configuration
2. Compile and boot the new kernel
3. mount -t container container -o memory /container
   will mount the memory controller to the /container mount point
4. mkdir /container/a
5. echo $$ > /container/a/tasks (add tasks to the new container)
6. echo -n <num_pages> > /container/a/memory.limit
   example
   echo -n 204800 > /container/a/memory.limit, sets the limit to 800 MB

on a system with 4K page size
7. run tasks, see the memory controller work
8. Report results, provide feedback
9. Develop/use new patches and go to step 1

Test Results

Results for version 3 of the patch were posted at
http://lwn.net/Articles/242554/

The code was also tested on a power box with regular machine usage scenarios,
the config disabled and with a stress suite that touched all the memory
in the system and was limited in a container.

Documentation

An article describing the design of the memory controller is available
at http://lwn.net/Articles/243795/

Observations

You might find some pages left over after all tasks have exited the
container. Even a sync followed by echo 1 > /proc/sys/vm/drop_caches
will not clean up all pages. The pages left behind are swap cache pages.
This problem can be easily solved by switching accounting to just
mapped pages. A mechanism to force all memory out of a container is
under investigation.

series

res_counters_infra.patch
mem-control-setup.patch
mem-control-accounting-setup.patch
mem-control-accounting.patch
mem-control-task-migration.patch
mem-control-lru-and-reclaim.patch
mem-control-out-of-memory.patch
mem-control-choose-rss-vs-rss-and-pagecache.patch
mem-control-per-container-page-referenced.patch

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

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

Subject: [-mm PATCH 1/9] Memory controller resource counters (v5)
Posted by Balbir Singh on Mon, 13 Aug 2007 17:41:23 GMT
View Forum Message <> Reply to Message

From: Pavel Emelianov <xemul@openvz.org>

Introduce generic structures and routines for resource accounting.

Each resource accounting container is supposed to aggregate it,
container_subsystem_state and its resource-specific members within.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>
Signed-off-by: <balbir@linux.vnet.ibm.com>
---

include/linux/res_counter.h | 102 ++
init/Kconfig | 7 ++
kernel/Makefile | 1
kernel/res_counter.c | 120 ++
4 files changed, 230 insertions(+)

diff -puN /dev/null include/linux/res_counter.h
--- /dev/null 2007-06-01 20:42:04.000000000 +0530
+++ linux-2.6.23-rc1-mm1-balbir/include/linux/res_counter.h 2007-08-13 23:06:11.000000000 +0530
@@ -0,0 +1,102 @@
+#ifndef __RES_COUNTER_H__
+#define __RES_COUNTER_H__
+
+/*
+ * Resource Counters
+ * Contain common data types and routines for resource accounting
+ *
+ * Copyright 2007 OpenVZ SWsoft Inc
+ *
+ * Author: Pavel Emelianov <xemul@openvz.org>
+ *
+ */
+
+#include <linux/container.h>
+
+/*
+ * The core object. the container that wishes to account for some
+ * resource may include this counter into its structures and use
+ * the helpers described beyond

```
+ */
+
+struct res_counter {
+ /*
+  * the current resource consumption level
+  */
+ unsigned long usage;
+ /*
+  * the limit that usage cannot exceed
+  */
+ unsigned long limit;
+ /*
+  * the number of unsuccessful attempts to consume the resource
+  */
+ unsigned long failcnt;
+ /*
+  * the lock to protect all of the above.
+  * the routines below consider this to be IRQ-safe
+  */
+ spinlock_t lock;
+};
+
+/*
+ * Helpers to interact with userspace
+ * res_counter_read/_write - put/get the specified fields from the
+ * res_counter struct to/from the user
+ *
+ * @counter:     the counter in question
+ * @member:  the field to work with (see RES_xxx below)
+ * @buf:     the buffer to opeate on,...
+ * @nbytes:  its size...
+ * @pos:     and the offset.
+ */
+
+ssize_t res_counter_read(struct res_counter *counter, int member,
+  const char __user *buf, size_t nbytes, loff_t *pos);
+ssize_t res_counter_write(struct res_counter *counter, int member,
+  const char __user *buf, size_t nbytes, loff_t *pos);
+
+/*
+ * the field descriptors. one for each member of res_counter
+ */
+
+enum {
+ RES_USAGE,
+ RES_LIMIT,
+ RES_FAILCNT,
+};
```

```
+
+/*
+ * helpers for accounting
+ */
+
+void res_counter_init(struct res_counter *counter);
+
+/*
+ * charge - try to consume more resource.
+ *
+ * @counter: the counter
+ * @val: the amount of the resource. each controller defines its own
+ *       units, e.g. numbers, bytes, Kbytes, etc
+ *
+ * returns 0 on success and <0 if the counter->usage will exceed the
+ * counter->limit _locked call expects the counter->lock to be taken
+ */
+
+int res_counter_charge_locked(struct res_counter *counter, unsigned long val);
+int res_counter_charge(struct res_counter *counter, unsigned long val);
+
+/*
+ * uncharge - tell that some portion of the resource is released
+ *
+ * @counter: the counter
+ * @val: the amount of the resource
+ *
+ * these calls check for usage underflow and show a warning on the console
+ * _locked call expects the counter->lock to be taken
+ */
+
+void res_counter_uncharge_locked(struct res_counter *counter, unsigned long val);
+void res_counter_uncharge(struct res_counter *counter, unsigned long val);
+
+#endif
diff -puN init/Kconfig~res_counters_infra init/Kconfig
--- linux-2.6.23-rc1-mm1/init/Kconfig~res_counters_infra 2007-08-13 23:06:11.000000000 +0530
+++ linux-2.6.23-rc1-mm1-balbir/init/Kconfig 2007-08-13 23:06:11.000000000 +0530
@@ -321,6 +321,13 @@ config CPUSETS

    Say N if unsure.

+config RESOURCE_COUNTERS
+ bool "Resource counters"
+ help
+   This option enables controller independent resource accounting
+         infrastructure that works with containers
+ depends on CONTAINERS
```

```
+
 config SYSFS_DEPRECATED
  bool "Create deprecated sysfs files"
  default y
diff -puN kernel/Makefile~res_counters_infra kernel/Makefile
--- linux-2.6.23-rc1-mm1/kernel/Makefile~res_counters_infra 2007-08-13 23:06:11.000000000
+0530
+++ linux-2.6.23-rc1-mm1-balbir/kernel/Makefile 2007-08-13 23:06:11.000000000 +0530
@@ -57,6 +57,7 @@ obj-$(CONFIG_RELAY) += relay.o
 obj-$(CONFIG_SYSCTL) += utsname_sysctl.o
 obj-$(CONFIG_TASK_DELAY_ACCT) += delayacct.o
 obj-$(CONFIG_TASKSTATS) += taskstats.o tsacct.o
+obj-$(CONFIG_RESOURCE_COUNTERS) += res_counter.o

 ifneq ($(CONFIG_SCHED_NO_NO_OMIT_FRAME_POINTER),y)
 # According to Alan Modra <alan@linuxcare.com.au>, the -fno-omit-frame-pointer is
diff -puN /dev/null kernel/res_counter.c
--- /dev/null 2007-06-01 20:42:04.000000000 +0530
+++ linux-2.6.23-rc1-mm1-balbir/kernel/res_counter.c 2007-08-13 23:06:11.000000000 +0530
@@ -0,0 +1,120 @@
+/*
+ * resource containers
+ *
+ * Copyright 2007 OpenVZ SWsoft Inc
+ *
+ * Author: Pavel Emelianov <xemul@openvz.org>
+ *
+ */
+
+#include <linux/types.h>
+#include <linux/parser.h>
+#include <linux/fs.h>
+#include <linux/res_counter.h>
+#include <linux/uaccess.h>
+
+void res_counter_init(struct res_counter *counter)
+{
+ spin_lock_init(&counter->lock);
+ counter->limit = (unsigned long)LONG_MAX;
+}
+
+int res_counter_charge_locked(struct res_counter *counter, unsigned long val)
+{
+ if (counter->usage > (counter->limit - val)) {
+  counter->failcnt++;
+  return -ENOMEM;
+ }
+
```

```
+ counter->usage += val;
+ return 0;
+}
+
+int res_counter_charge(struct res_counter *counter, unsigned long val)
+{
+ int ret;
+ unsigned long flags;
+
+ spin_lock_irqsave(&counter->lock, flags);
+ ret = res_counter_charge_locked(counter, val);
+ spin_unlock_irqrestore(&counter->lock, flags);
+ return ret;
+}
+
+void res_counter_uncharge_locked(struct res_counter *counter, unsigned long val)
+{
+ if (WARN_ON(counter->usage < val))
+  val = counter->usage;
+
+ counter->usage -= val;
+}
+
+void res_counter_uncharge(struct res_counter *counter, unsigned long val)
+{
+ unsigned long flags;
+
+ spin_lock_irqsave(&counter->lock, flags);
+ res_counter_uncharge_locked(counter, val);
+ spin_unlock_irqrestore(&counter->lock, flags);
+}
+
+
+static inline unsigned long *res_counter_member(struct res_counter *counter,
+      int member)
+{
+ switch (member) {
+ case RES_USAGE:
+  return &counter->usage;
+ case RES_LIMIT:
+  return &counter->limit;
+ case RES_FAILCNT:
+  return &counter->failcnt;
+ };
+
+ BUG();
+ return NULL;
+}
```

```
+
+ssize_t res_counter_read(struct res_counter *counter, int member,
+  const char __user *userbuf, size_t nbytes, loff_t *pos)
+{
+ unsigned long *val;
+ char buf[64], *s;
+
+ s = buf;
+ val = res_counter_member(counter, member);
+ s += sprintf(s, "%lu\n", *val);
+ return simple_read_from_buffer((void __user *)userbuf, nbytes,
+   pos, buf, s - buf);
+}
+
+ssize_t res_counter_write(struct res_counter *counter, int member,
+  const char __user *userbuf, size_t nbytes, loff_t *pos)
+{
+ int ret;
+ char *buf, *end;
+ unsigned long tmp, *val;
+
+ buf = kmalloc(nbytes + 1, GFP_KERNEL);
+ ret = -ENOMEM;
+ if (buf == NULL)
+  goto out;
+
+ buf[nbytes] = '\0';
+ ret = -EFAULT;
+ if (copy_from_user(buf, userbuf, nbytes))
+  goto out_free;
+
+ ret = -EINVAL;
+ tmp = simple_strtoul(buf, &end, 10);
+ if (*end != '\0')
+  goto out_free;
+
+ val = res_counter_member(counter, member);
+ *val = tmp;
+ ret = nbytes;
+out_free:
+ kfree(buf);
+out:
+ return ret;
+}
```

–

--
Warm Regards,

Balbir Singh
Linux Technology Center
IBM, ISTL

_____

Subject: [-mm PATCH 2/9] Memory controller containers setup (v5)
Posted by Balbir Singh on Mon, 13 Aug 2007 17:41:34 GMT
View Forum Message <> Reply to Message

Changelong
1. use depends instead of select in init/Kconfig
2. Port to v11
3. Clean up the usage of names (container files) for v11

Setup the memory container and add basic hooks and controls to integrate
and work with the container.

Signed-off-by: <balbir@linux.vnet.ibm.com>
---

 include/linux/container_subsys.h | 6 +
 include/linux/memcontrol.h | 21 ++++++
 init/Kconfig | 7 ++
 mm/Makefile | 1
 mm/memcontrol.c | 127 +++++++++++++++++++++++++++++++++++++++
 5 files changed, 162 insertions(+)

diff -puN include/linux/container_subsys.h~mem-control-setup include/linux/container_subsys.h
--- linux-2.6.23-rc1-mm1/include/linux/container_subsys.h~mem-control-setup 2007-08-13
23:06:11.000000000 +0530
+++ linux-2.6.23-rc1-mm1-balbir/include/linux/container_subsys.h 2007-08-13
23:06:11.000000000 +0530
@@ -30,3 +30,9 @@ SUBSYS(ns)
 #endif

 /* */
+
+#ifdef CONFIG_CONTAINER_MEM_CONT
+SUBSYS(mem_container)
+#endif
+
+/* */
diff -puN /dev/null include/linux/memcontrol.h
--- /dev/null 2007-06-01 20:42:04.000000000 +0530

```
+++ linux-2.6.23-rc1-mm1-balbir/include/linux/memcontrol.h 2007-08-13 23:06:11.000000000
+0530
@@ -0,0 +1,21 @@
+/* memcontrol.h - Memory Controller
+ *
+ * Copyright IBM Corporation, 2007
+ * Author Balbir Singh <balbir@linux.vnet.ibm.com>
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License as published by
+ * the Free Software Foundation; either version 2 of the License, or
+ * (at your option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ * GNU General Public License for more details.
+ */
+
+#ifndef _LINUX_MEMCONTROL_H
+#define _LINUX_MEMCONTROL_H
+
+#endif /* _LINUX_MEMCONTROL_H */
+
diff -puN init/Kconfig~mem-control-setup init/Kconfig
--- linux-2.6.23-rc1-mm1/init/Kconfig~mem-control-setup 2007-08-13 23:06:11.000000000 +0530
+++ linux-2.6.23-rc1-mm1-balbir/init/Kconfig 2007-08-13 23:06:11.000000000 +0530
@@ -357,6 +357,13 @@ config CONTAINER_NS
          for instance virtual servers and checkpoint/restart
          jobs.

+config CONTAINER_MEM_CONT
+ bool "Memory controller for containers"
+ depends on CONTAINERS && RESOURCE_COUNTERS
+ help
+   Provides a memory controller that manages both page cache and
+   RSS memory.
+
 config PROC_PID_CPUSET
   bool "Include legacy /proc/<pid>/cpuset file"
   depends on CPUSETS
diff -puN mm/Makefile~mem-control-setup mm/Makefile
--- linux-2.6.23-rc1-mm1/mm/Makefile~mem-control-setup 2007-08-13 23:06:11.000000000
+0530
+++ linux-2.6.23-rc1-mm1-balbir/mm/Makefile 2007-08-13 23:06:11.000000000 +0530
@@ -30,4 +30,5 @@ obj-$(CONFIG_FS_XIP) += filemap_xip.o
 obj-$(CONFIG_MIGRATION) += migrate.o
 obj-$(CONFIG_SMP) += allocpercpu.o
```

```
 obj-$(CONFIG_QUICKLIST) += quicklist.o
+obj-$(CONFIG_CONTAINER_MEM_CONT) += memcontrol.o

diff -puN /dev/null mm/memcontrol.c
--- /dev/null 2007-06-01 20:42:04.000000000 +0530
+++ linux-2.6.23-rc1-mm1-balbir/mm/memcontrol.c 2007-08-13 23:06:11.000000000 +0530
@@ -0,0 +1,127 @@
+/* memcontrol.c - Memory Controller
+ *
+ * Copyright IBM Corporation, 2007
+ * Author Balbir Singh <balbir@linux.vnet.ibm.com>
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License as published by
+ * the Free Software Foundation; either version 2 of the License, or
+ * (at your option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ * GNU General Public License for more details.
+ */
+
+#include <linux/res_counter.h>
+#include <linux/memcontrol.h>
+#include <linux/container.h>
+
+struct container_subsys mem_container_subsys;
+
+/*
+ * The memory controller data structure. The memory controller controls both
+ * page cache and RSS per container. We would eventually like to provide
+ * statistics based on the statistics developed by Rik Van Riel for clock-pro,
+ * to help the administrator determine what knobs to tune.
+ *
+ * TODO: Add a water mark for the memory controller. Reclaim will begin when
+ * we hit the water mark.
+ */
+struct mem_container {
+ struct container_subsys_state css;
+ /*
+  * the counter to account for memory usage
+  */
+ struct res_counter res;
+};
+
+/*
+ * A meta page is associated with every page descriptor. The meta page
```

```
+ * helps us identify information about the container
+ */
+struct meta_page {
+ struct list_head lru;  /* per container LRU list */
+ struct page *page;
+ struct mem_container *mem_container;
+};
+
+
+static inline
+struct mem_container *mem_container_from_cont(struct container *cont)
+{
+ return container_of(container_subsys_state(cont,
+    mem_container_subsys_id), struct mem_container,
+    css);
+}
+
+static ssize_t mem_container_read(struct container *cont, struct cftype *cft,
+   struct file *file, char __user *userbuf, size_t nbytes,
+   loff_t *ppos)
+{
+ return res_counter_read(&mem_container_from_cont(cont)->res,
+    cft->private, userbuf, nbytes, ppos);
+}
+
+static ssize_t mem_container_write(struct container *cont, struct cftype *cft,
+    struct file *file, const char __user *userbuf,
+    size_t nbytes, loff_t *ppos)
+{
+ return res_counter_write(&mem_container_from_cont(cont)->res,
+    cft->private, userbuf, nbytes, ppos);
+}
+
+static struct cftype mem_container_files[] = {
+ {
+  .name = "usage",
+  .private = RES_USAGE,
+  .read = mem_container_read,
+ },
+ {
+  .name = "limit",
+  .private = RES_LIMIT,
+  .write = mem_container_write,
+  .read = mem_container_read,
+ },
+ {
+  .name = "failcnt",
+  .private = RES_FAILCNT,
```

```
+  .read = mem_container_read,
+ },
+};
+
+static struct container_subsys_state *
+mem_container_create(struct container_subsys *ss, struct container *cont)
+{
+ struct mem_container *mem;
+
+ mem = kzalloc(sizeof(struct mem_container), GFP_KERNEL);
+ if (!mem)
+  return -ENOMEM;
+
+ res_counter_init(&mem->res);
+ return &mem->css;
+}
+
+static void mem_container_destroy(struct container_subsys *ss,
+    struct container *cont)
+{
+ kfree(mem_container_from_cont(cont));
+}
+
+static int mem_container_populate(struct container_subsys *ss,
+    struct container *cont)
+{
+ return container_add_files(cont, ss, mem_container_files,
+     ARRAY_SIZE(mem_container_files));
+}
+
+struct container_subsys mem_container_subsys = {
+ .name = "memory",
+ .subsys_id = mem_container_subsys_id,
+ .create = mem_container_create,
+ .destroy = mem_container_destroy,
+ .populate = mem_container_populate,
+ .early_init = 0,
+};
_
```

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

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