Subject: [RFC][PATCH 0/4[memcg shrinking usage.
Posted by KAMEZAWA Hiroyuki on Fri, 18 Jul 2008 10:31:23 GMT
View Forum Message <> Reply to Message

This email is just for dumping my queue before OLS.
(Not testd well. just for sharing idea.)

including following patches.

[1/4] ... res_counter_set_limit for -EBUSY.
[2/4] ... res_coutner_check_under_val.
[3/4] ... memcg set limit .
[4/4] ... memcg shrink usage. (NEW!)

A brief story for patch 4/4.

I have been writing background-reclaim support to memcg and used some kernel
threads. But...
1. using kernel thread have some complexty and add some amounts of codes.
2. there are another resource control method, soft-limit is proposed.
   (and will be others.)
3. wise load balancing between several groups will be hard.

So, using kernel thread have some problems. Andrew Morton suggested that
"Do that in user-land. We already have amounts of user-land-helper now."

Patch 4/4 just adds "shrink_me!" interface. A user land daemon can decrease
the usage of resource by kicking this. I like this idea. How do you think ?

Next problem is how to make interaction between userland and the kernel.
I'm now interested in "inotify". (But have to learn it more ;)

P.S.
I'll be completely offline until the end of July once I leave Japan.
Please ask/request/blame me directly at Ottawa if you want ;)

See you!
-Kame

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

Subject: [RFC][PATCH 1/4] res_counter set_limit and -EBUSY.
Posted by KAMEZAWA Hiroyuki on Fri, 18 Jul 2008 10:34:37 GMT

Add an interface to set limit. This is necessary to memory resource controller
because it shrinks usage at set limit.

(*) Other controller may not need this interface to shrink usage because
    shrinking is not necessary or impossible.

This is an enhancement.
named as res_counter-limit-change-ebusy.patch

Changelog v1->v2
  - fixed white space bug.

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

 include/linux/res_counter.h | 15 +++++++++++++++
 1 file changed, 15 insertions(+)

Index: linux-2.6.26-rc8-mm1/include/linux/res_counter.h
===
--- linux-2.6.26-rc8-mm1.orig/include/linux/res_counter.h
+++ linux-2.6.26-rc8-mm1/include/linux/res_counter.h
@@ -176,4 +176,19 @@ static inline bool res_counter_can_add(s
  return ret;
 }

+static inline int res_counter_set_limit(struct res_counter *cnt,
+ unsigned long long limit)
+{
+ unsigned long flags;
+ int ret = -EBUSY;
+
+ spin_lock_irqsave(&cnt->lock, flags);
+ if (cnt->usage < limit) {
+ cnt->limit = limit;
+ ret = 0;
+ }
+ spin_unlock_irqrestore(&cnt->lock, flags);
+ return ret;
+}
+
 #endif

Subject: [RFC][PATCH 2/4] res_counter check usage under val
Posted by KAMEZAWA Hiroyuki on Fri, 18 Jul 2008 10:35:30 GMT
View Forum Message <> Reply to Message

Add an interface to check usage is below "val"

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

```
 include/linux/res_counter.h |   13 ++++++++++++
 1 file changed, 13 insertions(+)

Index: mmtom-stamp-2008-07-15-15-39/include/linux/res_counter.h
===================================================================
--- mmtom-stamp-2008-07-15-15-39.orig/include/linux/res_counter.h
+++ mmtom-stamp-2008-07-15-15-39/include/linux/res_counter.h
@@ -191,4 +191,17 @@ static inline int res_counter_set_limit(
  return ret;
 }

+static inline int res_counter_check_under_val(struct res_counter *cnt,
+ unsigned long long val)
+{
+ unsigned long flags;
+ int ret = 0;
+
+ spin_lock_irqsave(&cnt->flags, flags);
+ if (cnt->usage < val)
+  ret = 1;
+ spin_unlock_irqrestore(&cnt->flags, flags);
+ return ret;
+}
+
 #endif
```

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

Subject: [RFC][PATCH 3/4] memcg shrink usage at limit change
Posted by KAMEZAWA Hiroyuki on Fri, 18 Jul 2008 10:36:34 GMT
View Forum Message <> Reply to Message

Shrinking memory usage at limit change.

This is an enhancement. based on res_counter-limit-change-ebusy.patch

Changelog: v3 -> v4
 - core logic is separated into two parts. (reuse the function later.)
 - added cond_resched() again.
   1. I was pointed out that alloc_pages() does cond_resched() after
      try_to_free_pages().
   2. This routine is called by an user. A user can set 'nice' in general.

Changelog: v2 -> v3
 - supported interrupt by signal. (A user can stop limit change by Ctrl-C.)

Changelog: v1 -> v2
 - adjusted to be based on write_string() patch set
 - removed backword goto.
 - removed unneccesary cond_resched().

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

 Documentation/controllers/memory.txt | 3 -
 mm/memcontrol.c | 73 ++++++++++++++++++++++++++++++++++--
 2 files changed, 70 insertions(+), 6 deletions(-)

Index: mmtom-stamp-2008-07-15-15-39/mm/memcontrol.c
===
--- mmtom-stamp-2008-07-15-15-39.orig/mm/memcontrol.c
+++ mmtom-stamp-2008-07-15-15-39/mm/memcontrol.c
@@ -804,6 +804,55 @@ int mem_cgroup_shrink_usage(struct mm_st
 }

 /*
+ * Shrink routine works cooperatively with users rather than the kernel.
+ * So,
+ *   - we have to check signals
+ *   - we pass GFP_HIGHUSERMOVABLE.
+ * TODO?:
+ *   - allow timeout
+ */
+static int mem_cgroup_shrink_usage_to(struct mem_cgroup *memcg,
+ unsigned long long val)
+{
+ int retry_count = MEM_CGROUP_RECLAIM_RETRIES;
+ int progress;
+ int ret;
+
+ while (!res_counter_check_under_val(&memcg->res, val)) {
+ if (signal_pending(current)) {
+   ret = -EINTR;
+   break;

```
+  }
+  if (!retry_count) {
+   ret = -EBUSY;
+   break;
+  }
+  progress = try_to_free_mem_cgroup_pages(memcg,
+      GFP_HIGHUSER_MOVABLE);
+  if (!progress)
+   retry_count--;
+  cond_resched();
+ }
+ return 0;
+}
+
+int mem_cgroup_resize_limit(struct mem_cgroup *memcg, unsigned long long val)
+{
+ int ret = 0;
+
+ do  {
+  ret = mem_cgroup_shrink_usage_to(memcg, val);
+  if (ret < 0)
+   break;
+  /* memory usage shrinking succeed. */
+  if (res_counter_set_limit(&memcg->res, val))
+   break;
+ } while (1);
+ return ret;
+}
+
+
+/*
  * This routine traverse page_cgroup in given list and drop them all.
  * *And* this routine doesn't reclaim page itself, just removes page_cgroup.
  */
@@ -883,13 +932,29 @@ static u64 mem_cgroup_read(struct cgroup
  return res_counter_read_u64(&mem_cgroup_from_cont(cont)->res,
        cft->private);
 }
-
+/*
+ * The user of this function is...
+ * RES_LIMIT.
+ */
 static int mem_cgroup_write(struct cgroup *cont, struct cftype *cft,
       const char *buffer)
 {
- return res_counter_write(&mem_cgroup_from_cont(cont)->res,
-     cft->private, buffer,
```

```
-     res_counter_memparse_write_strategy);
+ struct mem_cgroup *memcg = mem_cgroup_from_cont(cont);
+ unsigned long long val;
+ int ret;
+
+ switch (cft->private) {
+ case RES_LIMIT:
+  /* This function does all necessary parse...reuse it */
+  ret = res_counter_memparse_write_strategy(buffer, &val);
+  if (!ret)
+   ret = mem_cgroup_resize_limit(memcg, val);
+  break;
+ default:
+  ret = -EINVAL; /* should be BUG() ? */
+  break;
+ }
+ return ret;
 }

 static int mem_cgroup_reset(struct cgroup *cont, unsigned int event)
Index: mmtom-stamp-2008-07-15-15-39/Documentation/controllers/memory.txt
===================================================================
--- mmtom-stamp-2008-07-15-15-39.orig/Documentation/controllers/memory.txt
+++ mmtom-stamp-2008-07-15-15-39/Documentation/controllers/memory.txt
@@ -242,8 +242,7 @@ rmdir() if there are no tasks.
 1. Add support for accounting huge pages (as a separate controller)
 2. Make per-cgroup scanner reclaim not-shared pages first
 3. Teach controller to account for shared-pages
-4. Start reclamation when the limit is lowered
-5. Start reclamation in the background when the limit is
+4. Start reclamation in the background when the limit is
    not yet hit but the usage is getting closer

 Summary
```

_____

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

---

Subject: [RFC][PATCH 4/4] memcg shrinking usage
Posted by KAMEZAWA Hiroyuki on Fri, 18 Jul 2008 10:37:54 GMT
View Forum Message <> Reply to Message

Andrew Morton suggested me "If you want to add background reclaim
per memcg, do that in user-land. Don't add kernel thread at el more."

In general, when we try to implement some automatic rich control to memcg,
there are two choices.
  1. do in the kernel.
  2. do in the user.

In these days, memory subsystem is getting larger and lareger and
"Do something compliated" in the kernel is not good manner.
And Linux has some amount of helper programs in userland for years.

This patch adds a core interface to implement a user daemon program
which controles memcg.

 - memory.shrink_in_bytes.

When a user want to shrink memory.usage below 400M, he can write

 # echo 400M > memory.shrink_in_bytes

TODO:
 - add a notifier interface to trigger daemon's action.
 - add asynchronous mode....is difficult..maybe the daemon has to use
   thread or fork to do control memcg in asynchronous manner.

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

 Documentation/controllers/memory.txt | 20 ++++++++++++++++---
 mm/memcontrol.c | 20 ++++++++++++++++++++
 2 files changed, 37 insertions(+), 3 deletions(-)

Index: mmtom-stamp-2008-07-15-15-39/mm/memcontrol.c
===
--- mmtom-stamp-2008-07-15-15-39.orig/mm/memcontrol.c
+++ mmtom-stamp-2008-07-15-15-39/mm/memcontrol.c
@@ -40,6 +40,16 @@ struct cgroup_subsys mem_cgroup_subsys _
 static struct kmem_cache *page_cgroup_cache __read_mostly;
 #define MEM_CGROUP_RECLAIM_RETRIES 5

+
+#define MEMCG_FILETAG_BASE 0xab00
+
+enum {
+ PRIVATE_FILETAG_SHRINK,
+ PRIVATE_FILETAG_MAX,
+};
+
+#define MEMCG_FILETAG_SHRINK (MEMCG_FILETAG_BASE +
PRIVATE_FILETAG_SHRINK)
+

```
 /*
  * Statistics for memory cgroup.
  */
@@ -950,6 +960,11 @@ static int mem_cgroup_write(struct cgrou
   if (!ret)
    ret = mem_cgroup_resize_limit(memcg, val);
   break;
+ case MEMCG_FILETAG_SHRINK:
+  ret = res_counter_memparse_write_strategy(buffer, &val):
+  if (!ret)
+   ret= mem_cgroup_shrink_usage_to(memcg, val);
+  break;
  default:
   ret = -EINVAL; /* should be BUG() ? */
   break;
@@ -1061,6 +1076,11 @@ static struct cftype mem_cgroup_files[]
   .name = "stat",
   .read_map = mem_control_stat_show,
  },
+ {
+  .name = "shrink_in_bytes",
+  .private = MEMCG_FILETAG_SHRINK,
+  .write_string = mem_cgroup_write,
+ },
 };

 static int alloc_mem_cgroup_per_zone_info(struct mem_cgroup *mem, int node)
Index: mmtom-stamp-2008-07-15-15-39/Documentation/controllers/memory.txt
===================================================================
--- mmtom-stamp-2008-07-15-15-39.orig/Documentation/controllers/memory.txt
+++ mmtom-stamp-2008-07-15-15-39/Documentation/controllers/memory.txt
@@ -152,17 +152,17 @@ The memory controller uses the following

 3. User Interface

-0. Configuration
+3.0 Configuration

 a. Enable CONFIG_CGROUPS
 b. Enable CONFIG_RESOURCE_COUNTERS
 c. Enable CONFIG_CGROUP_MEM_RES_CTLR

-1. Prepare the cgroups
+3.1 Prepare the cgroups
 # mkdir -p /cgroups
 # mount -t cgroup none /cgroups -o memory

-2. Make the new group and move bash into it
```

+3.2 Make the new group and move bash into it
 # mkdir /cgroups/0
 # echo $$ > /cgroups/0/tasks

@@ -196,6 +196,7 @@ this file after a write to guarantee the
 The memory.failcnt field gives the number of times that the cgroup limit was
 exceeded.

+3.4 force_empty
 The memory.stat file gives accounting information. Now, the number of
 caches, RSS and Active pages/Inactive pages are shown.

@@ -205,6 +206,19 @@ The memory.force_empty gives an interfac

 will drop all charges in cgroup. Currently, this is maintained for test.

+3.5 shrink_in_bytes
+A user can try to reclaim memory under a group. If you want to shrink the
+memory usage to below 400M,
+
+# echo 400M > memory.shrink_in_bytes
+
+The kernel tries to shrink memory usage to be under 400M.
+
+This is a kind of hard-to-use operation by hand but this is for group management
+softwares. They enables background page reclaim, and well-scheduled load
+balancing between groups.
+If it is hard to shrink usage, the kernel returns -EBUSY.
+
 4. Testing

 Balbir posted lmbench, AIM9, LTP and vmmstress results [10] and [11].

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

---

Subject: Re: [RFC][PATCH 1/4] res_counter set_limit and -EBUSY.
Posted by Pavel Emelianov on Mon, 21 Jul 2008 17:32:10 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> Add an interface to set limit. This is necessary to memory resource controller
> because it shrinks usage at set limit.
>
> (*) Other controller may not need this interface to shrink usage because

>   shrinking is not necessary or impossible.
>
> This is an enhancement.
> named as res_counter-limit-change-ebusy.patch
>
> Changelog v1->v2
>   - fixed white space bug.
>
> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

Acked-by: Pavel Emelyanov <xemul@openvz.org>

>
> include/linux/res_counter.h | 15 +++++++++++++++
> 1 file changed, 15 insertions(+)
>
> Index: linux-2.6.26-rc8-mm1/include/linux/res_counter.h
> ===
> --- linux-2.6.26-rc8-mm1.orig/include/linux/res_counter.h
> +++ linux-2.6.26-rc8-mm1/include/linux/res_counter.h
> @@ -176,4 +176,19 @@ static inline bool res_counter_can_add(s
>   return ret;
> }
>
> +static inline int res_counter_set_limit(struct res_counter *cnt,
> + unsigned long long limit)
> +{
> + unsigned long flags;
> + int ret = -EBUSY;
> +
> + spin_lock_irqsave(&cnt->lock, flags);
> + if (cnt->usage < limit) {
> + cnt->limit = limit;
> + ret = 0;
> + }
> + spin_unlock_irqrestore(&cnt->lock, flags);
> + return ret;
> +}
> +
> #endif
>
>

Subject: Re: [RFC][PATCH 2/4] res_counter check usage under val
Posted by Pavel Emelianov on Mon, 21 Jul 2008 17:41:35 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> Add an interface to check usage is below "val"
>
> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
>
> include/linux/res_counter.h | 13 ++++++++++++
> 1 file changed, 13 insertions(+)
>
> Index: mmtom-stamp-2008-07-15-15-39/include/linux/res_counter.h
> ===
> --- mmtom-stamp-2008-07-15-15-39.orig/include/linux/res_counter.h
> +++ mmtom-stamp-2008-07-15-15-39/include/linux/res_counter.h
> @@ -191,4 +191,17 @@ static inline int res_counter_set_limit(
>   return ret;
> }
>
> +static inline int res_counter_check_under_val(struct res_counter *cnt,
> + unsigned long long val)
> +{
> + unsigned long flags;
> + int ret = 0;
> +
> + spin_lock_irqsave(&cnt->flags, flags);

Is this spilock protection *really* required? As far as I see
from its usage it is racy itself wrt to res_counter update, so
this locking looks superfluous.

> + if (cnt->usage < val)
> + ret = 1;
> + spin_unlock_irqrestore(&cnt->flags, flags);
> + return ret;
> +}
> +
> #endif
>
>

_____

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