Subject: [RFC][for -mm] memory controller enhancements for reclaiming take2 [0/8] introduction
Posted by KAMEZAWA Hiroyuki on Mon, 03 Dec 2007 09:33:55 GMT
View Forum Message <> Reply to Message

Hi, here is my memory cgroup patchset for 2.6.24-rc3-mm2.
(for review again. sorry. added -mm to the CC:)

Patch contents are
- clean up
- (possible) race fix.
- throttling LRU scan.
- background reclaim and high/low watermark

If it's better to divide aboves into a few sets, I'll do so.

Tested on ia64/8CPU/NUMA and x86_64/2CPU/SMP.

[Patches]
[1/8] ... remove unused variable (clean up)
[2/8] ... change 'if' sentense to BUG_ON() (clean up)
[3/8] ... add free_mem_cgroup_per_zone_info (clean up)
[4/8] ... possible race fix in resource controler
[5/8] ... throttling simultaneous direct reclaim under cgroup
[6/8] ... high/low watermark for resource controler
[7/8] ... use high/low watermark in memory controler
[8/8] ... wake up reclaim waiters at unchage.

TODO(1): Should I care resource controler users who doesn't use watermarks ?
and how should I do ?

==

here is a brief kernbench result on ia64/8CPU/2-node NUMA.
(average of 3 runs)

Used 800M limitation and High/Low watermark is 790M/760M (for patched kernel.)
make -j 4 and make -j 32.

*** Before patch ***********

Average Half Load -j 4 Run:	Average Optimal -j 32 Load Run:
Elapsed Time 266.104	Elapsed Time 353.957
User Time 1015.82	User Time 1070.68
System Time 70.0701	System Time 154.63
Percent CPU 407.667	Percent CPU 351.667
Context Switches 136916	Context Switches 223173
Sleeps 135404	Sleeps 199366

*** After patch ***********
Average Half Load -j 4 Run:
Elapsed Time 266.96
User Time 1016.55
System Time 70.24
Percent CPU 406.666
Context Switches 133219
Sleeps 137232

Average Optimal -j 32 Load Run:
Elapsed Time 232.32 ---(*1)
User Time 1120.9
System Time 116.843 ---(*2)
Percent CPU 537.667 ---(*3)
Context Switches 246752
Sleeps 195215

make -j 4 result has no difference between "before" and "after".
(800M was enough)
make -j 32 result has a difference.

(*1) Elapsed Time is decreased.
(*2) System Time is decreased.
(*3) CPU Utilization is increased.

This is because %iowait is decreased and "recaliming too much" is avoided.

here is vmstat -n 60 result amount make -j 32.

*** Before Patch ***

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs    us sy id wa st
37  6 416176 5758384  47824 288656   25 15823   311 18408 37259 2935  31  5 39 25  0
10 28 590336 5774528  50112 294560  323 27815   950 28225 52751 3405  45  8  4 43  0
18 28 648384 5614352  53744 298464  507 27710  1148 28226 54039 3521  60  9  2 29  0
 2 38 921312 5269648  50256 285392  185 29883   995 30471 55296 3000  39  7  8 46  0
 4  9 974944 5710672  52288 309104  206 31675   950 31885 54836 3074  48  6  6 40  0
 4 33 919568 5553984  49520 285664  119 3775   891  6850 17000 2810   21  4 63 12  0
15 24 1063856 5290144  51904 294128   56 27886   952 28406 56501 3317 54  9  1 36  0
 0 44 1399696 4933296  54592 292416  260 28046   847 28361 58778 3104 43  7  5 46  0
 0 47 1351200 4992208  56400 302896  275 22439   816 22694 44746 2945 32  5 14 50  0
 2 40 1433568 4878784  58032 290768  189 19617   971 19907 33764 2883 20  4 17 59  0
 3 41 1600608 4716688  59168 304496  155 19675   598 20032 39205 3007 39  4 14 43  0
 0  0 1143728 5587440  81456 301792  373 4040  1095  7052 11288 3006  30  3 50 17  0
 0 56 1487536 5281472  52208 276624   52 21253   403 21568 42882 2793 31  6 18 45  0
.....
```

*** After Patch ***

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs     us sy id wa st
25 22 271872 5869728  51792 290656   14 12589   593 13659 68145 3692   68  8 20  3  0
30 25 546592 5621552  53216 288128  106 25837  1058 26541 98549 3377   70  9  8 12  0
 1 16 878384 5103024  54064 290032  345 25477  1251 26176 94491 3325   67  8  8 18  0
 1  0 767200 5918896  82032 302704   25 8035  1023 10953 32435 2859    30  4 56 10  0
28 10 948640 5156912  53920 290416   21 13784   814 14706 77824 3740   84  9  5  3  0
```

 0 15 1320224 4683840 55680 296912 158 28465   893 29008 98158 3303 65 8 8 19 0
40 15 1619600 4372128 55424 297424 136 31633   848 32190 102808 3188 60 7 6 27 0
 0 0 1529152 5203728 83584 299968   47 11217   950 14145 39845 2799 27 4 52 17 0
30 10 1821776 4272416 56384 292528   16 13483   831 14271 77710 3749 77 9 11 3 0
33 16 2016464 3998480 56448 296416   96 19846   739 20454 78992 3342 58 6 11 26 0
 0 17 2148832 3820112 56544 289920   68 19842   528 20350 75083 2966 45 5 23 26 0
30 17 2118320 3891520 56320 293776   44 17544   533 18000 61345 2613 36 4 23 36 0

%iowait is decreased to some extent.
%user is increased (as a result)

No meaningful difference when memory is enough.

Thanks,
-Kame

Subject: [RFC][for -mm] memory controller enhancements for reclaiming take2 [1/8] clean up : remove unused va
Posted by KAMEZAWA Hiroyuki on Mon, 03 Dec 2007 09:35:37 GMT
View Forum Message <> Reply to Message

This check is not necessary now.

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

Index: linux-2.6.24-rc3-mm2/mm/memcontrol.c
===
--- linux-2.6.24-rc3-mm2.orig/mm/memcontrol.c
+++ linux-2.6.24-rc3-mm2/mm/memcontrol.c

```
@@ -860,9 +860,7 @@ retry:
   /* Avoid race with charge */
   atomic_set(&pc->ref_cnt, 0);
   if (clear_page_cgroup(page, pc) == pc) {
-   int active;
    css_put(&mem->css);
-   active = pc->flags & PAGE_CGROUP_FLAG_ACTIVE;
    res_counter_uncharge(&mem->res, PAGE_SIZE);
    __mem_cgroup_remove_list(pc);
    kfree(pc);
```

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

---

Subject: [RFC][for -mm] memory controller enhancements for reclaiming take2 [2/8] add BUG_ON() in mem_cgroup_
Posted by KAMEZAWA Hiroyuki on Mon, 03 Dec 2007 09:36:12 GMT
View Forum Message <> Reply to Message

This should be BUG_ON(). I misunderstood initialization path.

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

```
 mm/memcontrol.c |    3 +--
 1 file changed, 1 insertion(+), 2 deletions(-)

Index: linux-2.6.24-rc3-mm2/mm/memcontrol.c
===================================================================
--- linux-2.6.24-rc3-mm2.orig/mm/memcontrol.c
+++ linux-2.6.24-rc3-mm2/mm/memcontrol.c
@@ -206,8 +206,7 @@ static void mem_cgroup_charge_statistics
 static inline struct mem_cgroup_per_zone *
 mem_cgroup_zoneinfo(struct mem_cgroup *mem, int nid, int zid)
 {
- if (!mem->info.nodeinfo[nid])
-  return NULL;
+ BUG_ON(!mem->info.nodeinfo[nid]);
  return &mem->info.nodeinfo[nid]->zoneinfo[zid];
 }
```

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

Subject: [RFC][for -mm] memory controller enhancements for reclaiming take2 [3/8] define free_mem_cgroup_per_
Posted by KAMEZAWA Hiroyuki on Mon, 03 Dec 2007 09:37:19 GMT
View Forum Message <> Reply to Message

Now allocation of per_zone of mem_controller is done by
alloc_mem_cgroup_per_zone_info(). Then it will be good to use
free_mem_cgroup_per_zone_info() for maintainance.

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

```
 mm/memcontrol.c |    9 +++++++--
 1 file changed, 7 insertions(+), 2 deletions(-)

Index: linux-2.6.24-rc3-mm2/mm/memcontrol.c
===================================================================
--- linux-2.6.24-rc3-mm2.orig/mm/memcontrol.c
+++ linux-2.6.24-rc3-mm2/mm/memcontrol.c
@@ -1141,6 +1141,11 @@ static int alloc_mem_cgroup_per_zone_inf
  return 0;
 }

+static void free_mem_cgroup_per_zone_info(struct mem_cgroup *mem, int node)
+{
+ kfree(mem->info.nodeinfo[node]);
+}
+

 static struct mem_cgroup init_mem_cgroup;

@@ -1171,7 +1176,7 @@ mem_cgroup_create(struct cgroup_subsys *
  return &mem->css;
 free_out:
  for_each_node_state(node, N_POSSIBLE)
-  kfree(mem->info.nodeinfo[node]);
+  free_mem_cgroup_per_zone_info(mem, node);
  if (cont->parent != NULL)
   kfree(mem);
  return NULL;
@@ -1191,7 +1196,7 @@ static void mem_cgroup_destroy(struct cg
  struct mem_cgroup *mem = mem_cgroup_from_cont(cont);

  for_each_node_state(node, N_POSSIBLE)
-  kfree(mem->info.nodeinfo[node]);
+  free_mem_cgroup_per_zone_info(mem, node);

  kfree(mem_cgroup_from_cont(cont));
 }
```

Subject: [RFC][for -mm] memory controller enhancements for reclaiming take2 [5/8] throttling simultaneous cal
Posted by KAMEZAWA Hiroyuki on Mon, 03 Dec 2007 09:38:04 GMT
View Forum Message <> Reply to Message

Add throttling direct reclaim.

Trying heavy workload under memory controller, you'll see too much
iowait and system seems heavy. (This is not good.... memory controller
is usually used for isolating system workload)
And too much memory are reclaimed.

This patch adds throttling function for direct reclaim.
Currently, num_online_cpus/(4) + 1 threads can do direct memory reclaim
under memory controller.

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

```
 mm/memcontrol.c |   42 ++++++++++++++++++++++++++++++++++++++++++-
 1 file changed, 41 insertions(+), 1 deletion(-)

Index: linux-2.6.24-rc3-mm2/mm/memcontrol.c
===================================================================
--- linux-2.6.24-rc3-mm2.orig/mm/memcontrol.c
+++ linux-2.6.24-rc3-mm2/mm/memcontrol.c
@@ -36,6 +36,10 @@
 struct cgroup_subsys mem_cgroup_subsys;
 static const int MEM_CGROUP_RECLAIM_RETRIES = 5;

+/* Page reclaim throttle parameter */
+#define MEM_CGROUP_THROTTLE_RECLAIM_FACTOR (4)
+
+
 /*
  * Statistics for memory cgroup.
  */
@@ -133,6 +137,8 @@ struct mem_cgroup {

  unsigned long control_type; /* control RSS or RSS+Pagecache */
  int prev_priority; /* for recording reclaim priority */
```

```
+ atomic_t reclaimers; /* # of processes which calls reclaim */
+ wait_queue_head_t waitq;
  /*
   * statistics.
   */
@@ -565,6 +571,27 @@ unsigned long mem_cgroup_isolate_pages(u
  return nr_taken;
 }

+static void mem_cgroup_wait_reclaim(struct mem_cgroup *mem)
+{
+ DEFINE_WAIT(wait);
+ while (1) {
+  prepare_to_wait(&mem->waitq, &wait, TASK_INTERRUPTIBLE);
+  if (res_counter_check_under_limit(&mem->res)) {
+   finish_wait(&mem->waitq, &wait);
+   break;
+  }
+  schedule();
+  finish_wait(&mem->waitq, &wait);
+ }
+}
+
+/* throttle simlutaneous LRU scan */
+static int mem_cgroup_throttle_reclaim(struct mem_cgroup *mem)
+{
+ int limit = num_online_cpus()/MEM_CGROUP_THROTTLE_RECLAIM_FACTOR + 1;
+ return atomic_add_unless(&mem->reclaimers, 1, limit);
+}
+
 /*
  * Charge the memory controller for page usage.
  * Return
@@ -635,14 +662,24 @@ retry:
   */
  while (res_counter_charge(&mem->res, PAGE_SIZE)) {
   bool is_atomic = gfp_mask & GFP_ATOMIC;
+  int  ret;
   /*
    * We cannot reclaim under GFP_ATOMIC, fail the charge
    */
   if (is_atomic)
    goto noreclaim;

-  if (try_to_free_mem_cgroup_pages(mem, gfp_mask))
+  if (mem_cgroup_throttle_reclaim(mem)) {
+   ret = try_to_free_mem_cgroup_pages(mem, gfp_mask);
+   atomic_dec(&mem->reclaimers);
```

```
+   if (waitqueue_active(&mem->waitq))
+    wake_up_all(&mem->waitq);
+   if (ret)
+    continue;
+  } else {
+   mem_cgroup_wait_reclaim(mem);
    continue;
+  }

   /*
    * try_to_free_mem_cgroup_pages() might not give us a full
@@ -1169,6 +1206,9 @@ mem_cgroup_create(struct cgroup_subsys *
  mem->control_type = MEM_CGROUP_TYPE_ALL;
  memset(&mem->info, 0, sizeof(mem->info));

+ atomic_set(&mem->reclaimers, 0);
+ init_waitqueue_head(&mem->waitq);
+
  for_each_node_state(node, N_POSSIBLE)
   if (alloc_mem_cgroup_per_zone_info(mem, node))
    goto free_out;
```

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

---

Subject: [RFC][for -mm] memory controller enhancements for reclaiming take2 [4/8] possible race fix in res_co

Posted by KAMEZAWA Hiroyuki on Mon, 03 Dec 2007 09:38:09 GMT
View Forum Message <> Reply to Message

spinlock is necessary when someone changes res->counter value.
splited out from YAMAMOTO's background page reclaim for memory cgroup set.

Changelog v1 -> v2:
 - fixed type of "flags".

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
From: YAMAMOTO Takashi <yamamoto@valinux.co.jp>

```
 kernel/res_counter.c |    4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

Index: linux-2.6.24-rc3-mm2/kernel/res_counter.c
```

```
==========================================================================
--- linux-2.6.24-rc3-mm2.orig/kernel/res_counter.c
+++ linux-2.6.24-rc3-mm2/kernel/res_counter.c
@@ -98,6 +98,7 @@ ssize_t res_counter_write(struct res_cou
 {
  int ret;
  char *buf, *end;
+ unsigned long flags;
  unsigned long long tmp, *val;

  buf = kmalloc(nbytes + 1, GFP_KERNEL);
@@ -121,9 +122,10 @@ ssize_t res_counter_write(struct res_cou
   if (*end != '\0')
    goto out_free;
  }
-
+ spin_lock_irqsave(&counter->lock, flags);
  val = res_counter_member(counter, member);
  *val = tmp;
+ spin_unlock_irqrestore(&counter->lock, flags);
  ret = nbytes;
 out_free:
  kfree(buf);
```

Subject: [RFC][for -mm] memory controller enhancements for reclaiming take2 [6/8] high_low watermark for res_

Posted by KAMEZAWA Hiroyuki on Mon, 03 Dec 2007 09:40:50 GMT

View Forum Message <> Reply to Message

This patch adds high/low watermark parameter to res_counter.
and check routine.

splitted out from YAMAMOTO's background page reclaim for memory cgroup set.

TODO?
 - if res_counter's user doesn't want high/low watermark, res_counter_write()
   should ignore low <= high <= limit limitation ?

Changelog
 * added param watermark_state this can ba read without lock lock.

```
Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
From: YAMAMOTO Takashi <yamamoto@valinux.co.jp>

 include/linux/res_counter.h |   28 ++++++++++++++++++++++++++++
 kernel/res_counter.c        |   42 ++++++++++++++++++++++++++++++++++++++++++-
 2 files changed, 69 insertions(+), 1 deletion(-)

Index: linux-2.6.24-rc3-mm2/include/linux/res_counter.h
===================================================================
--- linux-2.6.24-rc3-mm2.orig/include/linux/res_counter.h
+++ linux-2.6.24-rc3-mm2/include/linux/res_counter.h
@@ -19,6 +19,12 @@
  * the helpers described beyond
  */

+enum watermark_state {
+ RES_WMARK_BELOW_LOW,
+ RES_WMARK_ABOVE_LOW,
+ RES_WMARK_ABOVE_HIGH,
+};
+
 struct res_counter {
  /*
   * the current resource consumption level
@@ -33,10 +39,17 @@ struct res_counter {
   */
  unsigned long long failcnt;
  /*
+  * Watermarks. Must keep low <= high <= limit.
+  */
+ unsigned long long high_watermark;
+ unsigned long long low_watermark;
+ /*
   * the lock to protect all of the above.
   * the routines below consider this to be IRQ-safe
   */
  spinlock_t lock;
+ /* can be read without lock */
+ enum watermark_state watermark_state;
 };

 /*
@@ -66,6 +79,8 @@ enum {
  RES_USAGE,
  RES_LIMIT,
  RES_FAILCNT,
+ RES_HWMARK,
+ RES_LWMARK,
```

```
 };

 /*
@@ -124,4 +139,17 @@ static inline bool res_counter_check_und
  return ret;
 }

+/*
+ * Helper function for implementing high/low watermark to resource controller.
+ */
+static inline bool res_counter_below_low_watermark(struct res_counter *cnt)
+{
+ return (cnt->watermark_state == RES_WMARK_BELOW_LOW);
+}
+
+static inline bool res_counter_above_high_watermark(struct res_counter *cnt)
+{
+ return (cnt->watermark_state == RES_WMARK_ABOVE_HIGH);
+}
+
 #endif
Index: linux-2.6.24-rc3-mm2/kernel/res_counter.c
===================================================================
--- linux-2.6.24-rc3-mm2.orig/kernel/res_counter.c
+++ linux-2.6.24-rc3-mm2/kernel/res_counter.c
@@ -17,6 +17,9 @@ void res_counter_init(struct res_counter
 {
  spin_lock_init(&counter->lock);
  counter->limit = (unsigned long long)LLONG_MAX;
+ counter->low_watermark = (unsigned long long)LLONG_MAX;
+ counter->high_watermark = (unsigned long long)LLONG_MAX;
+ counter->watermark_state = RES_WMARK_BELOW_LOW;
 }

 int res_counter_charge_locked(struct res_counter *counter, unsigned long val)
@@ -27,6 +30,12 @@ int res_counter_charge_locked(struct res
  }

  counter->usage += val;
+
+ if (counter->usage > counter->high_watermark)
+  counter->watermark_state = RES_WMARK_ABOVE_HIGH;
+ else if (counter->usage > counter->low_watermark)
+  counter->watermark_state = RES_WMARK_ABOVE_LOW;
+
  return 0;
 }
```

```
@@ -47,6 +56,11 @@ void res_counter_uncharge_locked(struct
   val = counter->usage;

  counter->usage -= val;
+
+ if (counter->usage < counter->low_watermark)
+  counter->watermark_state = RES_WMARK_BELOW_LOW;
+ else if (counter->usage < counter->high_watermark)
+  counter->watermark_state = RES_WMARK_ABOVE_LOW;
 }

 void res_counter_uncharge(struct res_counter *counter, unsigned long val)
@@ -69,6 +83,10 @@ res_counter_member(struct res_counter *c
   return &counter->limit;
  case RES_FAILCNT:
   return &counter->failcnt;
+ case RES_HWMARK:
+  return &counter->high_watermark;
+ case RES_LWMARK:
+  return &counter->low_watermark;
  };

  BUG();
@@ -123,12 +141,34 @@ ssize_t res_counter_write(struct res_cou
    goto out_free;
  }
  spin_lock_irqsave(&counter->lock, flags);
+ /*
+  * check low <= high <= limit.
+  */
+ switch (member) {
+  case RES_LIMIT:
+   if (counter->high_watermark > tmp)
+    goto unlock_free;
+   break;
+  case RES_HWMARK:
+   if (tmp < counter->low_watermark ||
+       tmp > counter->limit)
+    goto unlock_free;
+   break;
+  case RES_LWMARK:
+   if (tmp > counter->high_watermark)
+    goto unlock_free;
+   break;
+  default:
+   break;
+ }
  val = res_counter_member(counter, member);
```

```
  *val = tmp;
- spin_unlock_irqrestore(&counter->lock, flags);
  ret = nbytes;
+unlock_free:
+ spin_unlock_irqrestore(&counter->lock, flags);
 out_free:
  kfree(buf);
 out:
  return ret;
 }
+
```

Subject: [RFC][for -mm] memory controller enhancements for reclaiming take2 [7/8] bacground reclaim for memor
Posted by KAMEZAWA Hiroyuki on Mon, 03 Dec 2007 09:42:44 GMT
View Forum Message <> Reply to Message

High Low watermark for page reclaiming in memory cgroup(1) and background
reclaim routine.

 - If USAGE is bigger than high watermark, background reclaim starts.
 - If USAGE is lower than low watermark, background reclaim stops.

Each value is represented in bytes (See by kernel/res_counter.c)

Changelog v1 -> v2:
  - merged high/low patch and background reclaim patch.
  - removed automatic adjustement of high/low value.
  - cleanups.
  - add schedule_timedwait() in background reclaim's busy loop.
    (LRU scan can be very heavy workload and cosume CPUS.)

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
From: YAMAMOTO Takashi <yamamoto@valinux.co.jp>

 mm/memcontrol.c | 97 ++---
 1 file changed, 92 insertions(+), 5 deletions(-)

Index: linux-2.6.24-rc3-mm2/mm/memcontrol.c
===
--- linux-2.6.24-rc3-mm2.orig/mm/memcontrol.c

```
+++ linux-2.6.24-rc3-mm2/mm/memcontrol.c
@@ -30,6 +30,8 @@
 #include <linux/spinlock.h>
 #include <linux/fs.h>
 #include <linux/seq_file.h>
+#include <linux/kthread.h>
+#include <linux/freezer.h>

 #include <asm/uaccess.h>

@@ -117,11 +119,6 @@ struct mem_cgroup_lru_info {
  * page cache and RSS per cgroup. 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. May be even add a low water mark, such that
- * no reclaim occurs from a cgroup at it's low water mark, this is
- * a feature that will be implemented much later in the future.
  */
 struct mem_cgroup {
  struct cgroup_subsys_state css;
@@ -130,6 +127,13 @@ struct mem_cgroup {
   */
  struct res_counter res;
  /*
+  * background reclaim
+  */
+ struct {
+  wait_queue_head_t waitq;
+  struct task_struct *thread;
+ } daemon;
+ /*
   * Per cgroup active and inactive list, similar to the
   * per zone LRU lists.
   */
@@ -401,6 +405,17 @@ static void __mem_cgroup_move_lists(stru
  }
 }

+static void
+mem_cgroup_schedule_reclaim(struct mem_cgroup *mem)
+{
+ if (!unlikely(mem->daemon.thread))
+  return;
+ if (!waitqueue_active(&mem->daemon.waitq))
+  return;
+ wake_up_interruptible(&mem->daemon.waitq);
```

```
+}
+
+
 int task_in_mem_cgroup(struct task_struct *task, const struct mem_cgroup *mem)
 {
  int ret;
@@ -708,6 +723,8 @@ noreclaim:
    mem_cgroup_out_of_memory(mem, GFP_KERNEL);
   goto free_pc;
  }
+ if (res_counter_above_high_watermark(&mem->res))
+  mem_cgroup_schedule_reclaim(mem);

  atomic_set(&pc->ref_cnt, 1);
  pc->mem_cgroup = mem;
@@ -863,6 +880,42 @@ retry:
 }

 /*
+ * Background page reclaim daeom for memory controller.
+ */
+
+static int mem_cgroup_reclaim_daemon(void *data)
+{
+ DEFINE_WAIT(wait);
+ struct mem_cgroup *mem = data;
+
+ css_get(&mem->css);
+ current->flags |= PF_SWAPWRITE;
+ set_freezable();
+
+ while (!kthread_should_stop()) {
+  prepare_to_wait(&mem->daemon.waitq, &wait, TASK_INTERRUPTIBLE);
+  if (res_counter_below_low_watermark(&mem->res)) {
+   if (!kthread_should_stop()) {
+    schedule();
+    try_to_freeze();
+   }
+   finish_wait(&mem->daemon.waitq, &wait);
+   continue;
+  }
+  finish_wait(&mem->daemon.waitq, &wait);
+  try_to_free_mem_cgroup_pages(mem, GFP_HIGHUSER_MOVABLE);
+  /*
+   * throttle LRU scan for moderate reclaiming.
+   * not congestion wait.
+   */
+  schedule_timeout(HZ/10);
```

```
+ }
+
+ css_put(&mem->css);
+ return 0;
+}
+
+/*
  * This routine traverse page_cgroup in given list and drop them all.
  * This routine ignores page_cgroup->ref_cnt.
  * *And* this routine doesn't reclaim page itself, just removes page_cgroup.
@@ -1136,6 +1189,18 @@ static struct cftype mem_cgroup_files[]
   .read = mem_control_type_read,
  },
  {
+  .name = "low_watermark_in_bytes",
+  .private = RES_LWMARK,
+  .write = mem_cgroup_write,
+  .read = mem_cgroup_read,
+ },
+ {
+  .name = "high_watermark_in_bytes",
+  .private = RES_HWMARK,
+  .write = mem_cgroup_write,
+  .read = mem_cgroup_read,
+ },
+ {
   .name = "force_empty",
   .write = mem_force_empty_write,
   .read = mem_force_empty_read,
@@ -1186,6 +1251,16 @@ static void free_mem_cgroup_per_zone_inf

 static struct mem_cgroup init_mem_cgroup;

+static int __init mem_cgroup_reclaim_init(void)
+{
+ init_mem_cgroup.daemon.thread = kthread_run(mem_cgroup_reclaim_daemon,
+     &init_mem_cgroup, "memcontd");
+ if (IS_ERR(init_mem_cgroup.daemon.thread))
+  BUG();
+ return 0;
+}
+late_initcall(mem_cgroup_reclaim_init);
+
 static struct cgroup_subsys_state *
 mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
 {
@@ -1213,6 +1288,17 @@ mem_cgroup_create(struct cgroup_subsys *
   if (alloc_mem_cgroup_per_zone_info(mem, node))
```

```
    goto free_out;

+ /* Memory Reclaim Daemon per cgroup */
+ init_waitqueue_head(&mem->daemon.waitq);
+ if (mem != &init_mem_cgroup) {
+  /* Complicated...but we cannot call kthread create here..*/
+  /* init call will later assign kthread */
+  mem->daemon.thread = kthread_run(mem_cgroup_reclaim_daemon,
+     mem, "memcontd");
+  if (IS_ERR(mem->daemon.thread))
+   goto free_out;
+ }
+
  return &mem->css;
 free_out:
  for_each_node_state(node, N_POSSIBLE)
@@ -1227,6 +1313,7 @@ static void mem_cgroup_pre_destroy(struc
 {
  struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
  mem_cgroup_force_empty(mem);
+ kthread_stop(mem->daemon.thread);
 }

 static void mem_cgroup_destroy(struct cgroup_subsys *ss,
```

Subject: [RFC][for -mm] memory controller enhancements for reclaiming take2 [8/8] wake up waiters at unchage

Posted by KAMEZAWA Hiroyuki on Mon, 03 Dec 2007 09:45:36 GMT

View Forum Message <> Reply to Message

Throttling direct reclaim reduces the sytem load. But waiters are only waken up if someone finish try_to_free_mem_cgroup_pages().

In progress of reclaiming, there can be enough memory before try_to_free_xxx is finished. Because we throttle the number of reclaimers, it's better to wake up waiters if there is enough room, in moderate way.
This decreases the system idle time under memory pressure in cgroup.

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

```
 mm/memcontrol.c |    7 +++++++
 1 file changed, 7 insertions(+)

Index: linux-2.6.24-rc3-mm2/mm/memcontrol.c
===================================================================
--- linux-2.6.24-rc3-mm2.orig/mm/memcontrol.c
+++ linux-2.6.24-rc3-mm2/mm/memcontrol.c
@@ -816,6 +816,13 @@ void mem_cgroup_uncharge(struct page_cgr
    __mem_cgroup_remove_list(pc);
    spin_unlock_irqrestore(&mz->lru_lock, flags);
    kfree(pc);
+   /*
+    * If there is enough room but there are waiters,
+    * wake up one. (wake up all is tend to be heavy)
+    */
+   if (!res_counter_above_high_watermark(&mem->res) &&
+       waitqueue_active(&mem->waitq))
+    wake_up(&mem->waitq);
   }
  }
 }
```

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [5/8] throttling simultaneous

Posted by Rik van Riel on Mon, 03 Dec 2007 14:24:18 GMT
View Forum Message <> Reply to Message

On Mon, 3 Dec 2007 18:39:21 +0900
KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:

> Add throttling direct reclaim.
>
> Trying heavy workload under memory controller, you'll see too much
> iowait and system seems heavy. (This is not good.... memory controller
> is usually used for isolating system workload)
> And too much memory are reclaimed.
>
> This patch adds throttling function for direct reclaim.
> Currently, num_online_cpus/(4) + 1 threads can do direct memory reclaim
> under memory controller.

The same problems are true of global reclaim.

Now that we're discussing this RFC anyway, I wonder if we
should think about moving this restriction to the global
reclaim level...

--
"Debugging is twice as hard as writing the code in the first place.
Therefore, if you write the code as cleverly as possible, you are,
by definition, not smart enough to debug it." - Brian W. Kernighan

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [5/8] throttling simultaneous
Posted by KAMEZAWA Hiroyuki on Tue, 04 Dec 2007 01:33:32 GMT
View Forum Message <> Reply to Message

On Mon, 3 Dec 2007 09:24:18 -0500
Rik van Riel <riel@redhat.com> wrote:

> On Mon, 3 Dec 2007 18:39:21 +0900
> KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:
>
> > Add throttling direct reclaim.
> >
> > Trying heavy workload under memory controller, you'll see too much
> > iowait and system seems heavy. (This is not good.... memory controller
> > is usually used for isolating system workload)
> > And too much memory are reclaimed.
> >
> > This patch adds throttling function for direct reclaim.
> > Currently, num_online_cpus/(4) + 1 threads can do direct memory reclaim
> > under memory controller.
>
> The same problems are true of global reclaim.
>
> Now that we're discussing this RFC anyway, I wonder if we
> should think about moving this restriction to the global
> reclaim level...
>
Hmm, I agree to some extent.
I'd like to add the same level of parameters to memory controller AMAP.

But, IMHO, there are differences basically.

Memory controller's reclaim is much heavier than global LRU because of
increasing footprint , the number of atomic ops....
And memory controller's reclaim policy is simpler than global because
it is not kicked by memory shortage and almost all gfk_mask is GFP_HIGHUSER_MOVABLE
and order is always 0.

I think starting from throttling memory controller is not so bad because
it's heavy and it's simple. The benefit of this throttoling is clearer than
globals.

Adding this kind of controls to global memory allocator/LRU may cause
unexpected slow down in application's response time. High-response application
users may dislike this. We may need another gfp_flag or sysctl to allow
throttling in global.
For memory controller, the user sets its memory limitation by himself. He can
adjust parameters and the workload. So, I think this throttoling is not so
problematic in memory controller as global.

Of course, we can export "do throttoling or not" control in cgroup interface.

Thanks,
-Kame

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [7/8] bacground reclaim for m
Posted by yamamoto on Tue, 04 Dec 2007 03:07:55 GMT
View Forum Message <> Reply to Message

> @@ -1186,6 +1251,16 @@ static void free_mem_cgroup_per_zone_inf
>
> static struct mem_cgroup init_mem_cgroup;
>
> +static int __init mem_cgroup_reclaim_init(void)
> +{
> + init_mem_cgroup.daemon.thread = kthread_run(mem_cgroup_reclaim_daemon,
> +   &init_mem_cgroup, "memcontd");
> + if (IS_ERR(init_mem_cgroup.daemon.thread))
> + BUG();
> + return 0;

> +}
> +late_initcall(mem_cgroup_reclaim_init);
> +
> static struct cgroup_subsys_state *
> mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
> {
> @@ -1213,6 +1288,17 @@ mem_cgroup_create(struct cgroup_subsys *
>   if (alloc_mem_cgroup_per_zone_info(mem, node))
>   goto free_out;
>
> + /* Memory Reclaim Daemon per cgroup */
> + init_waitqueue_head(&mem->daemon.waitq);
> + if (mem != &init_mem_cgroup) {
> + /* Complicated...but we cannot call kthread create here..*/
> + /* init call will later assign kthread */
> + mem->daemon.thread = kthread_run(mem_cgroup_reclaim_daemon,
> +   mem, "memcontd");
> + if (IS_ERR(mem->daemon.thread))
> +   goto free_out;
> + }
> +
>   return &mem->css;
> free_out:
>   for_each_node_state(node, N_POSSIBLE)

you don't need the kthread as far as RES_HWMARK is "infinite".
given the current default value of RES_HWMARK, you can simplify
initialization by deferring the kthread creation to mem_cgroup_write.

YAMAMOTO Takashi

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [7/8] bacground reclaim for m
Posted by KAMEZAWA Hiroyuki on Tue, 04 Dec 2007 03:18:49 GMT
View Forum Message <> Reply to Message

On Tue, 4 Dec 2007 12:07:55 +0900 (JST)
yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:
> you don't need the kthread as far as RES_HWMARK is "infinite".
> given the current default value of RES_HWMARK, you can simplify
> initialization by deferring the kthread creation to mem_cgroup_write.
>
Hmm, will try. But I wonder whether assumption can be true forever or not.

For example, when memory controller supports sub-group and a relationship
between a parent group and children groups are established.

But, think it later is an one way ;) I'll try to make things simpler.

Thanks,
-Kame

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [7/8] bacground reclaim for m
Posted by yamamoto on Tue, 04 Dec 2007 03:31:18 GMT
View Forum Message <> Reply to Message

> On Tue, 4 Dec 2007 12:07:55 +0900 (JST)
> yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:
> > you don't need the kthread as far as RES_HWMARK is "infinite".
> > given the current default value of RES_HWMARK, you can simplify
> > initialization by deferring the kthread creation to mem_cgroup_write.
> >
> Hmm, will try. But I wonder whether assumption can be true forever or not.
> For example, when memory controller supports sub-group and a relationship
> between a parent group and children groups are established.
>
> But, think it later is an one way ;) I'll try to make things simpler.
>
> Thanks,
> -Kame

the point is to create a thread when setting RES_HWMARK.
mem_cgroup_write is merely an example. it can be when inheriting
watermarks from the parent cgroup, etc.
anyway, the assumption we need here is that the default value of
the top level cgroup's high watermark is infinite. i think it's
a quite reasonable assumption.

YAMAMOTO Takashi

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [0/8] introduction
Posted by KAMEZAWA Hiroyuki on Tue, 04 Dec 2007 06:46:20 GMT
View Forum Message <> Reply to Message

On Mon, 3 Dec 2007 18:33:55 +0900
KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:

> [Patches]
> [1/8] ... remove unused variable (clean up)
> [2/8] ... change 'if' sentense to BUG_ON() (clean up)
> [3/8] ... add free_mem_cgroup_per_zone_info (clean up)
> [4/8] ... possible race fix in resource controler
> [5/8] ... throttling simultaneous direct reclaim under cgroup
> [6/8] ... high/low watermark for resource controler
> [7/8] ... use high/low watermark in memory controler
> [8/8] ... wake up reclaim waiters at unchage.
>

Hi, Andrew. I think patch 5-8 should be got more review and discussion.

I'd like to schedule them to the next -mm. (because rc4 is shipped...)
But patch 1-4 are just clean up and bug fix and seems no side-effect.

Could you pick up 1-4 to -mm if Balbir and Pavel gives me ack ?

Thanks,
-Kame

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

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [5/8] throttling simultaneous
Posted by Balbir Singh on Tue, 04 Dec 2007 13:27:22 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> On Mon, 3 Dec 2007 09:24:18 -0500
> Rik van Riel <riel@redhat.com> wrote:
>
>> On Mon, 3 Dec 2007 18:39:21 +0900
>> KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:
>>
>>> Add throttling direct reclaim.

>>>
>>> Trying heavy workload under memory controller, you'll see too much
>>> iowait and system seems heavy. (This is not good.... memory controller
>>> is usually used for isolating system workload)
>>> And too much memory are reclaimed.
>>>
>>> This patch adds throttling function for direct reclaim.
>>> Currently, num_online_cpus/(4) + 1 threads can do direct memory reclaim
>>> under memory controller.
>> The same problems are true of global reclaim.
>>
>> Now that we're discussing this RFC anyway, I wonder if we
>> should think about moving this restriction to the global
>> reclaim level...
>>
> Hmm, I agree to some extent.
> I'd like to add the same level of parameters to memory controller AMAP.
>

The CKRM memory controller had the following parameters for throttling

Watermarks

shrink_at
shrink_to

and

num_shrinks
shrink_interval

Number of times shrink can be called in a shrink_interval.

> But, IMHO, there are differences basically.
>
> Memory controller's reclaim is much heavier than global LRU because of
> increasing footprint , the number of atomic ops....
> And memory controller's reclaim policy is simpler than global because
> it is not kicked by memory shortage and almost all gfk_mask is GFP_HIGHUSER_MOVABLE
> and order is always 0.
>
> I think starting from throttling memory controller is not so bad because
> it's heavy and it's simple. The benefit of this throttoling is clearer than
> globals.
>

I think global throttling is good as well, sometimes under heavy load I

find several tasks stuck in reclaim. I suspect throttling them and avoid
this scenario. May be worth experimenting an thinking about it deserves
more discussion.

> Adding this kind of controls to global memory allocator/LRU may cause
> unexpected slow down in application's response time. High-response application
> users may dislike this. We may need another gfp_flag or sysctl to allow
> throttling in global.
> For memory controller, the user sets its memory limitation by himself. He can
> adjust parameters and the workload. So, I think this throttoling is not so
> problematic in memory controller as global.
>
> Of course, we can export "do throttoling or not" control in cgroup interface.
>

I think we should export the interface.

>
> Thanks,
> -Kame
>

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

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [0/8] introduction
Posted by Balbir Singh on Tue, 04 Dec 2007 14:25:05 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> Hi, here is my memory cgroup patchset for 2.6.24-rc3-mm2.
> (for review again. sorry. added -mm to the CC:)
>
> Patch contents are
> - clean up
> - (possible) race fix.
> - throttling LRU scan.
> - background reclaim and high/low watermark
>

> If it's better to divide aboves into a few sets, I'll do so.
>
> Tested on ia64/8CPU/NUMA and x86_64/2CPU/SMP.
>
> [Patches]
> [1/8] ... remove unused variable (clean up)
> [2/8] ... change 'if' sentense to BUG_ON() (clean up)
> [3/8] ... add free_mem_cgroup_per_zone_info (clean up)
> [4/8] ... possible race fix in resource controler
> [5/8] ... throttling simultaneous direct reclaim under cgroup
> [6/8] ... high/low watermark for resource controler
> [7/8] ... use high/low watermark in memory controler
> [8/8] ... wake up reclaim waiters at unchage.
>
> TODO(1): Should I care resource controler users who doesn't use watermarks ?
> and how should I do ?
>

It would be nice to have the watermark set to a good default value. Then
use ACL on the file memory.*watermark to control what users can control
this file.

> ==
>
> here is a brief kernbench result on ia64/8CPU/2-node NUMA.
> (average of 3 runs)
>
> Used 800M limitation and High/Low watermark is 790M/760M (for patched kernel.)
> make -j 4 and make -j 32.
>
>
> *** Before patch ***************
> Average Half Load -j 4 Run: Average Optimal -j 32 Load Run:
> Elapsed Time 266.104 Elapsed Time 353.957
> User Time 1015.82 User Time 1070.68
> System Time 70.0701 System Time 154.63
> Percent CPU 407.667 Percent CPU 351.667
> Context Switches 136916 Context Switches 223173
> Sleeps 135404 Sleeps 199366
>
> *** After patch ****************
> Average Half Load -j 4 Run: Average Optimal -j 32 Load Run:
> Elapsed Time 266.96 Elapsed Time 232.32 ---(*1)
> User Time 1016.55 User Time 1120.9
> System Time 70.24 System Time 116.843 ---(*2)
> Percent CPU 406.666 Percent CPU 537.667 ---(*3)
> Context Switches 133219 Context Switches 246752
> Sleeps 137232 Sleeps 195215

>
> make -j 4 result has no difference between "before" and "after".
> (800M was enough)
> make -j 32 result has a difference.
>
> (*1) Elapsed Time is decreased.
> (*2) System Time is decreased.
> (*3) CPU Utilization is increased.
>

KAMEZAWA-San, what happens if we use a little less aggressive set of
watermarks, something like

700/300

Can we keep the defaults something close to what each zone uses?
pages_low, pages_high and pages_min.

> This is because %iowait is decreased and "recaliming too much" is avoided.
>
> here is vmstat -n 60 result amount make -j 32.
>
> *** Before Patch ***
> procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
> r b swpd free buff cache si so bi bo in cs us sy id wa st
> 37 6 416176 5758384 47824 288656 25 15823 311 18408 37259 2935 31 5 39 25 0
> 10 28 590336 5774528 50112 294560 323 27815 950 28225 52751 3405 45 8 4 43 0
> 18 28 648384 5614352 53744 298464 507 27710 1148 28226 54039 3521 60 9 2 29 0
> 2 38 921312 5269648 50256 285392 185 29883 995 30471 55296 3000 39 7 8 46 0
> 4 9 974944 5710672 52288 309104 206 31675 950 31885 54836 3074 48 6 6 40 0
> 4 33 919568 5553984 49520 285664 119 3775 891 6850 17000 2810 21 4 63 12 0
> 15 24 1063856 5290144 51904 294128 56 27886 952 28406 56501 3317 54 9 1 36 0
> 0 44 1399696 4933296 54592 292416 260 28046 847 28361 58778 3104 43 7 5 46 0
> 0 47 1351200 4992208 56400 302896 275 22439 816 22694 44746 2945 32 5 14 50 0
> 2 40 1433568 4878784 58032 290768 189 19617 971 19907 33764 2883 20 4 17 59 0
> 3 41 1600608 4716688 59168 304496 155 19675 598 20032 39205 3007 39 4 14 43 0
> 0 0 1143728 5587440 81456 301792 373 4040 1095 7052 11288 3006 30 3 50 17 0
> 0 56 1487536 5281472 52208 276624 52 21253 403 21568 42882 2793 31 6 18 45 0
> .....
>
> *** After Patch ***
> procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
> r b swpd free buff cache si so bi bo in cs us sy id wa st
> 25 22 271872 5869728 51792 290656 14 12589 593 13659 68145 3692 68 8 20 3 0
> 30 25 546592 5621552 53216 288128 106 25837 1058 26541 98549 3377 70 9 8 12 0
> 1 16 878384 5103024 54064 290032 345 25477 1251 26176 94491 3325 67 8 8 18 0
> 1 0 767200 5918896 82032 302704 25 8035 1023 10953 32435 2859 30 4 56 10 0
> 28 10 948640 5156912 53920 290416 21 13784 814 14706 77824 3740 84 9 5 3 0

> 0 15 1320224 4683840 55680 296912 158 28465   893 29008 98158 3303 65 8 8 19 0
> 40 15 1619600 4372128 55424 297424 136 31633   848 32190 102808 3188 60 7 6 27 0
> 0 0 1529152 5203728 83584 299968   47 11217   950 14145 39845 2799 27 4 52 17 0
> 30 10 1821776 4272416 56384 292528   16 13483   831 14271 77710 3749 77 9 11 3 0
> 33 16 2016464 3998480 56448 296416   96 19846   739 20454 78992 3342 58 6 11 26 0
> 0 17 2148832 3820112 56544 289920   68 19842   528 20350 75083 2966 45 5 23 26 0
> 30 17 2118320 3891520 56320 293776   44 17544   533 18000 61345 2613 36 4 23 36 0
>
> %iowait is decreased to some extent.
> %user is increased (as a result)
>
> No meaningful difference when memory is enough.
>
> Thanks,
> -Kame
>
Thanks,

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

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [1/8] clean up : remove unuse
Posted by Balbir Singh on Tue, 04 Dec 2007 15:55:19 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> This check is not necessary now.
>
> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
>
>
> Index: linux-2.6.24-rc3-mm2/mm/memcontrol.c
> ===
> --- linux-2.6.24-rc3-mm2.orig/mm/memcontrol.c
> +++ linux-2.6.24-rc3-mm2/mm/memcontrol.c
> @@ -860,9 +860,7 @@ retry:
>   /* Avoid race with charge */
>   atomic_set(&pc->ref_cnt, 0);

> if (clear_page_cgroup(page, pc) == pc) {
> - int active;
> css_put(&mem->css);
> - active = pc->flags & PAGE_CGROUP_FLAG_ACTIVE;

Good Catch, __mem_cgroup_remove_list() takes care of this.

> res_counter_uncharge(&mem->res, PAGE_SIZE);
> __mem_cgroup_remove_list(pc);
> kfree(pc);
>

Acked-by: Balbir Singh <balbir@linux.vnet.ibm.com>

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

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [2/8] add BUG_ON() in mem_cgr
Posted by Balbir Singh on Tue, 04 Dec 2007 16:01:32 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> This should be BUG_ON(). I misunderstood initialization path.
>
> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
>
> mm/memcontrol.c | 3 +--
> 1 file changed, 1 insertion(+), 2 deletions(-)
>
> Index: linux-2.6.24-rc3-mm2/mm/memcontrol.c
> ===
> --- linux-2.6.24-rc3-mm2.orig/mm/memcontrol.c
> +++ linux-2.6.24-rc3-mm2/mm/memcontrol.c
> @@ -206,8 +206,7 @@ static void mem_cgroup_charge_statistics
> static inline struct mem_cgroup_per_zone *
> mem_cgroup_zoneinfo(struct mem_cgroup *mem, int nid, int zid)
> {
> - if (!mem->info.nodeinfo[nid])
> - return NULL;

> + BUG_ON(!mem->info.nodeinfo[nid]);
> return &mem->info.nodeinfo[nid]->zoneinfo[zid];
> }
>
>

Acked-by: Balbir Singh <balbir@linux.vnet.ibm.com>

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

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [3/8] define free_mem_cgroup_
Posted by Balbir Singh on Tue, 04 Dec 2007 16:32:05 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> Now allocation of per_zone of mem_controller is done by
> alloc_mem_cgroup_per_zone_info(). Then it will be good to use
> free_mem_cgroup_per_zone_info() for maintainance.
>
> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

Looks good

Acked-by: Balbir Singh <balbir@linux.vnet.ibm.com>

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

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2

[4/8] possible race fix in re
Posted by Balbir Singh on Tue, 04 Dec 2007 19:02:56 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> spinlock is necessary when someone changes res->counter value.
> splited out from YAMAMOTO's background page reclaim for memory cgroup set.
>
> Changelog v1 -> v2:
> - fixed type of "flags".
>
>
> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
> From: YAMAMOTO Takashi <yamamoto@valinux.co.jp>

Looks sane to me

Acked-by: Balbir Singh <balbir@linux.vnet.ibm.com>

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

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

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [5/8] throttling simultaneous
Posted by KAMEZAWA Hiroyuki on Wed, 05 Dec 2007 00:26:39 GMT
View Forum Message <> Reply to Message

On Tue, 04 Dec 2007 18:57:22 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> > Adding this kind of controls to global memory allocator/LRU may cause
> > unexpected slow down in application's response time. High-response application
> > users may dislike this. We may need another gfp_flag or sysctl to allow
> > throttling in global.
> > For memory controller, the user sets its memory limitation by himself. He can
> > adjust parameters and the workload. So, I think this throttoling is not so
> > problematic in memory controller as global.
> >
> > Of course, we can export "do throttoling or not" control in cgroup interface.
> >

>
> I think we should export the interface.
>
Ok, I'll export.

Thanks,
-Kame

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

---

Subject: Re: [RFC][for -mm] memory controller enhancements for reclaiming take2 [0/8] introduction
Posted by KAMEZAWA Hiroyuki on Wed, 05 Dec 2007 00:44:20 GMT
View Forum Message <> Reply to Message

On Tue, 04 Dec 2007 19:55:05 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
> KAMEZAWA-San, what happens if we use a little less aggressive set of
> watermarks, something like
>
> 700/300
>
will test today. you mean low=300M, high=700M, limit=800M case ?

> Can we keep the defaults something close to what each zone uses?
> pages_low, pages_high and pages_min.
>
After review of Pavel-san, "don't define *default* value" style is used here.
If we use default value, we'll have to detect "we should adjust high/low
watermarks when the limit changes."

That will complicate things and may crash the system administrators policy.
It's not havey work to adjust high/low limit to sutitable value (which was
defined against the workload by system admin) at setting limit.

Thanks,
-kame

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

---