
Subject: [PATCH 20/33] memory controller add documentation
Posted by [Paul Menage](#) on Mon, 17 Sep 2007 21:03:27 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Balbir Singh <balbir@linux.vnet.ibm.com>
(container->cgroup renaming by Paul Menage <menage@google.com>)

Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>
Signed-off-by: Paul Menage <menage@google.com>

Documentation/controllers/memory.txt | 259 ++++++
1 files changed, 259 insertions(+)

```
diff -puN /dev/null Documentation/controllers/memory.txt
--- /dev/null
+++ a/Documentation/controllers/memory.txt
@@ -0,0 +1,259 @@
+Memory Controller
+
+Salient features
+
+  +a. Enable control of both RSS (mapped) and Page Cache (unmapped) pages
+  +b. The infrastructure allows easy addition of other types of memory to control
+  +c. Provides *zero overhead* for non memory controller users
+  +d. Provides a double LRU: global memory pressure causes reclaim from the
+  +   global LRU; a cgroup on hitting a limit, reclaims from the per
+  +   cgroup LRU
+
+NOTE: Page Cache (unmapped) also includes Swap Cache pages as a subset
+and will not be referred to explicitly in the rest of the documentation.
+
+Benefits and Purpose of the memory controller
+
+The memory controller isolates the memory behaviour of a group of tasks
+from the rest of the system. The article on LWN [12] mentions some probable
+uses of the memory controller. The memory controller can be used to
+
+  +a. Isolate an application or a group of applications
+  +   Memory hungry applications can be isolated and limited to a smaller
+  +   amount of memory.
+  +b. Create a cgroup with limited amount of memory, this can be used
+  +   as a good alternative to booting with mem=XXXX.
+  +c. Virtualization solutions can control the amount of memory they want
+  +   to assign to a virtual machine instance.
+  +d. A CD/DVD burner could control the amount of memory used by the
+  +   rest of the system to ensure that burning does not fail due to lack
```

+ of available memory.

+e. There are several other use cases, find one or use the controller just for fun (to learn and hack on the VM subsystem).

+

+1. History

+

+The memory controller has a long history. A request for comments for the memory controller was posted by Balbir Singh [1]. At the time the RFC was posted there were several implementations for memory control. The goal of the RFC was to build consensus and agreement for the minimal features required for memory control. The first RSS controller was posted by Balbir Singh[2] in Feb 2007. Pavel Emelianov [3][4][5] has since posted three versions of the RSS controller. At OLS, at the resource management BoF, everyone suggested that we handle both page cache and RSS together. Another request was raised to allow user space handling of OOM. The current memory controller is at version 6; it combines both mapped (RSS) and unmapped Page Cache Control [11].

+

+2. Memory Control

+

+Memory is a unique resource in the sense that it is present in a limited amount. If a task requires a lot of CPU processing, the task can spread its processing over a period of hours, days, months or years, but with memory, the same physical memory needs to be reused to accomplish the task.

+

+The memory controller implementation has been divided into phases. These are:

+

- +1. Memory controller
- +2. mlock(2) controller
- +3. Kernel user memory accounting and slab control
- +4. user mappings length controller

+

+The memory controller is the first controller developed.

+

+2.1. Design

+

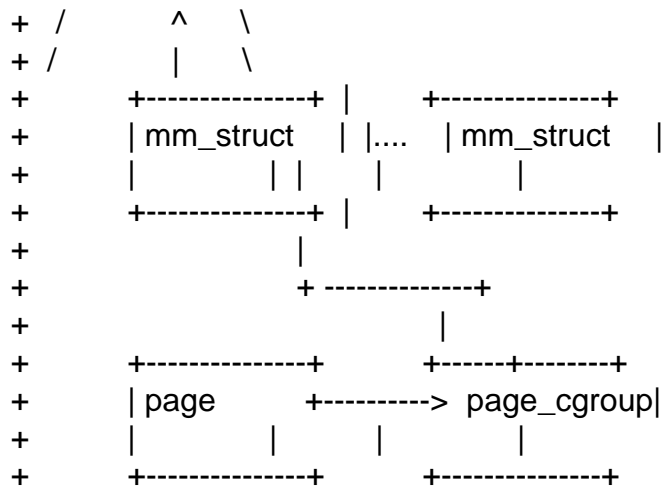
+The core of the design is a counter called the res_counter. The res_counter tracks the current memory usage and limit of the group of processes associated with the controller. Each cgroup has a memory controller specific data structure (mem_cgroup) associated with it.

+

+2.2. Accounting

+

```
+ +-----+
+ | mem_cgroup |
+ | (res_counter) |
+ +-----+
```



(Figure 1: Hierarchy of Accounting)

+Figure 1 shows the important aspects of the controller

- +1. Accounting happens per cgroup
- +2. Each mm_struct knows about which cgroup it belongs to
- +3. Each page has a pointer to the page_cgroup, which in turn knows the cgroup it belongs to

+The accounting is done as follows: mem_cgroup_charge() is invoked to setup the necessary data structures and check if the cgroup that is being charged is over its limit. If it is then reclaim is invoked on the cgroup.

+More details can be found in the reclaim section of this document.

+If everything goes well, a page meta-data-structure called page_cgroup is allocated and associated with the page. This routine also adds the page to the per cgroup LRU.

+2.2.1 Accounting details

+All mapped pages (RSS) and unmapped user pages (Page Cache) are accounted. RSS pages are accounted at the time of page_add_*_rmap() unless they've already been accounted for earlier. A file page will be accounted for as Page Cache; it's mapped into the page tables of a process, duplicate accounting is carefully avoided. Page Cache pages are accounted at the time of add_to_page_cache(). The corresponding routines that remove a page from the page tables or removes a page from Page Cache is used to decrement the accounting counters of the cgroup.

+2.3 Shared Page Accounting

+Shared pages are accounted on the basis of the first touch approach. The cgroup that first touches a page is accounted for the page. The principle behind this approach is that a cgroup that aggressively uses a shared

+page will eventually get charged for it (once it is uncharged from
+the cgroup that brought it in -- this will happen on memory pressure).

+

+2.4 Reclaim

+

+Each cgroup maintains a per cgroup LRU that consists of an active
+and inactive list. When a cgroup goes over its limit, we first try
+to reclaim memory from the cgroup so as to make space for the new
+pages that the cgroup has touched. If the reclaim is unsuccessful,
+an OOM routine is invoked to select and kill the bulkiest task in the
+cgroup.

+

+The reclaim algorithm has not been modified for cgroups, except that
+pages that are selected for reclaiming come from the per cgroup LRU
+list.

+

+2. Locking

+

+The memory controller uses the following hierarchy

+

- +1. zone->lru_lock is used for selecting pages to be isolated
- +2. mem->lru_lock protects the per cgroup LRU
- +3. lock_page_cgroup() is used to protect page->page_cgroup

+

+3. User Interface

+

+0. Configuration

+

- +a. Enable CONFIG_CGROUPS
- +b. Enable CONFIG_RESOURCE_COUNTERS
- +c. Enable CONFIG_CGROUP_MEM_CONT

+

+1. Prepare the cgroups

```
+# mkdir -p /cgroups
```

```
+# mount -t cgroup none /cgroups -o memory
```

+

+2. Make the new group and move bash into it

```
+# mkdir /cgroups/0
```

```
+# echo $$ > /cgroups/0/tasks
```

+

+Since now we're in the 0 cgroup,

+We can alter the memory limit:

```
+# echo -n 6000 > /cgroups/0/memory.limit
```

+

+We can check the usage:

```
+# cat /cgroups/0/memory.usage
```

+25

+

+The memory.failcnt field gives the number of times that the cgroup limit was exceeded.

+

+4. Testing

+

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

+Apart from that v6 has been tested with several applications and regular daily use. The controller has also been tested on the PPC64, x86_64 and UML platforms.

+

+4.1 Troubleshooting

+

+Sometimes a user might find that the application under a cgroup is terminated. There are several causes for this:

+

+1. The cgroup limit is too low (just too low to do anything useful)

+2. The user is using anonymous memory and swap is turned off or too low

+

+A sync followed by `echo 1 > /proc/sys/vm/drop_caches` will help get rid of some of the pages cached in the cgroup (page cache pages).

+

+4.2 Task migration

+

+When a task migrates from one cgroup to another, its charge is not carried forward. The pages allocated from the original cgroup still remain charged to it, the charge is dropped when the page is freed or reclaimed.

+

+4.3 Removing a cgroup

+

+A cgroup can be removed by `rmdir`, but as discussed in sections 4.1 and 4.2, a cgroup might have some charge associated with it, even though all tasks have migrated away from it. If some pages are still left, after following the steps listed in sections 4.1 and 4.2, check the Swap Cache usage in `/proc/meminfo` to see if the Swap Cache usage is showing up in the cgroups memory.usage counter. A simple test of `swapoff -a` and `swapon -a` should free any pending Swap Cache usage.

+

+4.4 Choosing what to account -- Page Cache (unmapped) vs RSS (mapped)?

+

+The type of memory accounted by the cgroup can be limited to just mapped pages by writing "1" to memory.control_type field

+

+`echo -n 1 > memory.control_type`

+

+5. TODO

+

+1. Add support for accounting huge pages (as a separate controller)

- +2. Improve the user interface to accept/display memory limits in KB or MB
 - + rather than pages (since page sizes can differ across platforms/machines).
- +3. Make cgroup lists per-zone
- +4. Make per-cgroup scanner reclaim not-shared pages first
- +5. Teach controller to account for shared-pages
- +6. Start reclamation when the limit is lowered
- +7. Start reclamation in the background when the limit is
 - + not yet hit but the usage is getting closer
- +8. Create per zone LRU lists per cgroup
- +
- +Summary
- +
- +Overall, the memory controller has been a stable controller and has been
 - +commented and discussed quite extensively in the community.
- +
- +References
- +
- +1. Singh, Balbir. RFC: Memory Controller, <http://lwn.net/Articles/206697/>
- +2. Singh, Balbir. Memory Controller (RSS Control),
 - + <http://lwn.net/Articles/222762/>
- +3. Emelianov, Pavel. Resource controllers based on process cgroups
 - + <http://lkml.org/lkml/2007/3/6/198>
- +4. Emelianov, Pavel. RSS controller based on process cgroups (v2)
 - + <http://lkml.org/lkml/2007/4/9/74>
- +5. Emelianov, Pavel. RSS controller based on process cgroups (v3)
 - + <http://lkml.org/lkml/2007/5/30/244>
- +6. Menage, Paul. Control Groups v10, <http://lwn.net/Articles/236032/>
- +7. Vaidyanathan, Srinivasan, Control Groups: Pagecache accounting and control
 - + subsystem (v3), <http://lwn.net/Articles/235534/>
- +8. Singh, Balbir. RSS controller V2 test results (lmbench),
 - + <http://lkml.org/lkml/2007/5/17/232>
- +9. Singh, Balbir. RSS controller V2 AIM9 results
 - + <http://lkml.org/lkml/2007/5/18/1>
- +10. Singh, Balbir. Memory controller v6 results,
 - + <http://lkml.org/lkml/2007/8/19/36>
- +11. Singh, Balbir. Memory controller v6, <http://lkml.org/lkml/2007/8/17/69>
- +12. Corbet, Jonathan, Controlling memory use in cgroups,
 - + <http://lwn.net/Articles/243795/>

—

--

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