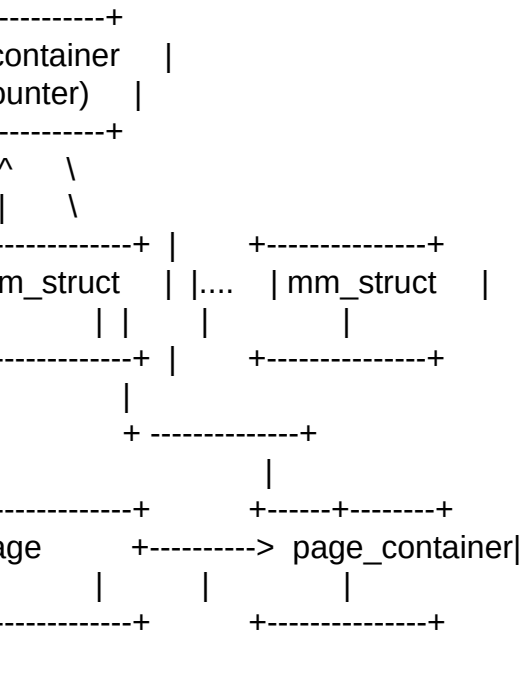
Subject: [PATCH] Memory controller Add Documentation
Posted by Balbir Singh on Wed, 22 Aug 2007 13:06:12 GMT
View Forum Message <> Reply to Message

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

Documentation/memcontrol.txt | 193 ++
1 file changed, 193 insertions(+)

diff -puN /dev/null Documentation/memcontrol.txt
--- /dev/null 2007-06-01 20:42:04.000000000 +0530
+++ linux-2.6.23-rc2-mm2-balbir/Documentation/memcontrol.txt 2007-08-22 18:29:29.000000000 +0530
@@ -0,0 +1,193 @@
+Memory Controller
+
+0. Salient features
+
+a. Enable control of both RSS and Page Cache pages
+b. The infrastructures 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 container on hitting a limit, reclaims from the per
+ container LRU
+
+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 RSS and 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 container has a memory controller specific data
+structure (mem_container) associated with it.
+
+2.2. Accounting
+
+ 
+
+
+ (Figure 1: Hierarchy of Accounting)
+
+
+Figure 1 shows the important aspects of the controller
+
+1. Accounting happens per container
+2. Each mm_struct knows about which container they belong to
+3. Each page has a pointer to the page_container, which in turn knows the
+ container it belongs to
+
+The accounting is done as follows, mem_container_charge() is invoked to setup
+the necessary data structures and check if the container that is being charged
+is over its limit. If it is then reclaim is invoked on the container.

+More details can be found in the reclaim section of this document.
+If everything goes well, a page meta-data-structure called page_container is
+allocated and associated with the page. This routine also adds the page to
+the per container LRU.
+
+2.3 Shared Page Accounting
+
+Shared pages are accounted on the basis of the first touch approach. The
+container that first touches a page is accounted for the page. The principle
+behind this approach is that a container that aggressively uses a shared
+page, will eventually get charged for it (once it is uncharged from
+the container that brought it in -- this will happen on memory pressure).
+
+2.4 Reclaim
+
+Each container maintains a per container LRU that consists of an active
+and inactive list. When a container goes over its limit, we first try
+and reclaim memory from the container so as to make space for the new
+pages that the container has touched. If the reclaim is unsuccessful,
+an OOM routine is invoked to select and kill the bulkiest task in the
+container.
+
+The reclaim algorithm has not been modified for containers, except that
+pages that are selected for reclaiming come from the per container LRU
+list.
+
+2.5
+
+3. User Interface
+
+0. Configuration
+
+a. Enable CONFIG_CONTAINERS
+b. Enable CONFIG_RESOURCE_COUNTERS
+c. Enable CONFIG_CONTAINER_MEM_CONT
+
+1. Prepare the containers
+# mkdir -p /containers
+# mount -t container none /containers -o memory
+
+2. Make the new group and move bash into it
+# mkdir /containers/0
+# echo $$ > /containers/0/tasks
+
+Since now we're in the 0 container.
+We can alter the memory limit
+# echo -n 6000 > /containers/0/memory.limit
+

+We can check the usage
+# cat /containers/0/memory.usage
+25
+
+The memory.failcnt gives the number of times that the container limit was
+exceeded.
+
+4. Testing
+
+Balbir posted lmbench, 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 container is
+terminated, there are several causes for this
+
+1. The container 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
+
+echo 1 > /proc/sys/vm/drop_pages will help get rid of some of the pages
+cached in the container (page cache pages).
+
+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 container lists per-zone
+4. Make per-container 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 container
+
+Summary
+
+Overall, the memory controller has been a stable controller and has been
+commented and discussed on 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 containers
+ http://lkml.org/lkml/2007/3/6/198
+4. Emelianov, Pavel. RSS controller based on process containers (v2)
+ http://lkml.org/lkml/2007/4/9/74
+5. Emelianov, Pavel. RSS controller based on process containers (v3)
+ http://lkml.org/lkml/2007/5/30/244
+6. Menage, Paul. Containers v10, http://lwn.net/Articles/236032/
+7. Vaidyanathan, Srinivasan, Containers: 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

_

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

_____