Subject: [PATCH -RSS] Add documentation for the RSS controller
Posted by Balbir Singh on Mon, 04 Jun 2007 17:50:25 GMT
View Forum Message <> Reply to Message

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

 Documentation/controller/rss.txt | 165 ++++++++++++++++++++++++++++++++++++++
 1 file changed, 165 insertions(+)

diff -puN /dev/null Documentation/controller/rss.txt
--- /dev/null 2007-06-01 20:42:04.000000000 +0530
+++ linux-2.6.22-rc2-mm1-balbir/Documentation/controller/rss.txt 2007-06-04 23:18:37.000000000 +0530
@@ -0,0 +1,165 @@
+RSS (Resident Set Size) controller
+
+1. History
+
+The RSS 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. The latest version works on top of v10 of the containers
+patches [6].
+
+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. RSS controller
+2. Page Cache controller
+3. mlock(2) controller
+4. Kernel user memory accounting and slab control
+
+The RSS controller is the first controller developed, the page cache controller
+is under development [7].
+
+2.1. Design
+

+The core of the design is a counter called the res_counter. The res_counter
+tracks the current RSS usage and limit of the group of processes associated
+with the controller. A res_counter is embedded in the mm_struct of a process
+and within the container that groups processes together. Each container
+has a RSS specific data structure (rss_container) associated with it.
+
+2.2. Accounting
+
+ +--------------------+
+ | container |
+ | (res_counter) |
+ +--------------------+
+ /   \
+ /   \
+   +---------------+   +---------------+
+ | mm_struct | .... | mm_struct |
+ | (res_counter) | | (res_counter) |
+   +---------------+   +---------------+
+
+   (Figure 1: Hierarchy of Accounting)
+
+
+Figure 1 shows two important aspects of the controller
+
+1. Accounting happens per mm_struct (per process)
+2. The accounting information of each mm_struct is accumulated in the container.
+
+(2) is required so that when a task migrates from container A to container B,
+the accounting of the task is known accurately and the charges can be
+carried over (*not done currently*) if desired.
+
+The accounting is done currently in two phases. In the first phase
+container_rss_prepare() 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.
+
+In the second phase, container_rss_add is invoked from page_add_*_rmap().
+This routine 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 (through isolate_pages_in_container())
+
+3. User Interface
+
+(From Pavel's posting)
+
+1. Prepare the containers
+# mkdir -p /containers/rss
+# mount -t container none /containers/rss -o rss
+
+2. Make the new group and move bash into it
+# mkdir /containers/rss/0
+# echo $$ > /containers/rss/0/tasks
+
+Since now we're in the 0 container.
+We can alter the RSS limit
+# echo -n 6000 > /containers/rss/0/rss_limit
+
+We can check the usage
+# cat /containers/rss/0/rss_usage
+25
+
+The rss_failcnt gives the number of times that the container limit was
+exceeded and the rss_reclaimed gives the count of the number of times
+reclaim was called.
+
+4. Testing
+
+Balbir posted lmbench [8] and AIM9 [9] results for the RSS v2[4] patches.
+Apart from that v2 has been tested with several applications for the OLS
+paper on memory control. These applications include web servers and database
+servers. RSS v2 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
+
+5. TODO
+
+1. Test v3 on more platforms and run more tests
+2. Add support for accounting huge pages (as a separate controller)
+3. Improve the user interface to accept/display memory limits in KB or MB
+ rather than pages (since page sizes can differ across platforms/machines).
+
+Summary
+
+Overall, the RSS 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

–

--
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