Subject: [-mm PATCH 0/9] Memory controller introduction (v6)
Posted by Balbir Singh on Fri, 17 Aug 2007 08:42:28 GMT
View Forum Message <> Reply to Message

Here's version 6 of the memory controller (against 2.6.23-rc2-mm2).

The tests that were run has been included in the _Test Results section_ below.

Changelog since version 5

1. Ported to 2.6.23-rc2-mm2
2. Added a css_put() in the case of race between allocation of page_containers
   (YAMAMOTO Takashi)

Changelog since version 4

1. Renamed meta_page to page_container (Nick Piggin)
2. Moved locking from page flags to last bit of the page_container pointer
   (Nick Piggin)
3. Fixed a rare race in mem_container_isolate_pages (YAMAMOTO Takashi)

Changelog since version 3

1. Ported to v11 of the containers patchset (2.6.23-rc1-mm1). Paul Menage
   helped immensely with a detailed review of v3
2. Reclaim is retried to allow reclaim of pages coming in as a result
   of mapped pages reclaim (swap cache growing as a result of RSS reclaim)
3. page_referenced() is now container aware. During container reclaim,
   references from other containers do not prevent a page from being
   reclaimed from a non-referencing container
4. Fixed a possible race condition spotted by YAMAMOTO Takashi

Changelog since version 2

1. Improved error handling in mm/memory.c (spotted by YAMAMOTO Takashi)
2. Test results included
3. try_to_free_mem_container_pages() bug fix (sc->may_writepage is now
   set to !laptop_mode)

Changelog since version 1

1. Fixed some compile time errors (in mm/migrate.c from Vaidyanathan S)
2. Fixed a panic seen when LIST_DEBUG is enabled
3. Added a mechanism to control whether we track page cache or both
   page cache and mapped pages (as requested by Pavel)
4. Dave Hansen provided detail review comments on the code.

This patchset implements another version of the memory controller. These

patches have been through a big churn, the first set of patches were posted
last year and earlier this year at
 http://lkml.org/lkml/2007/2/19/10

This patchset draws from the patches listed above and from some of the
contents of the patches posted by Vaidyanathan for page cache control.
 http://lkml.org/lkml/2007/6/20/92

At OLS, the resource management BOF, it was discussed that we need to manage
RSS and unmapped page cache together. This patchset is a step towards that

TODO's

1. Add memory controller water mark support. Reclaim on high water mark
2. Add support for shrinking on limit change
3. Add per zone per container LRU lists (this is being actively worked
   on by Pavel Emelianov)
4. Figure out a better CLUI for the controller
5. Add better statistics
6. Explore using read_unit64() as recommended by Paul Menage
   (NOTE: read_ulong() would also be nice to have)

In case you have been using/testing the RSS controller, you'll find that
this controller works slower than the RSS controller. The reason being
that both swap cache and page cache is accounted for, so pages do go
out to swap upon reclaim (they cannot live in the swap cache).

Any test output, feedback, comments, suggestions are welcome! I am committed
to fixing any bugs and improving the performance of the memory controller.
Do not hesitate to send any fixes, request for fixes that is required.

Using the patches

1. Enable Memory controller configuration
2. Compile and boot the new kernel
3. mount -t container container -o memory /container
   will mount the memory controller to the /container mount point
4. mkdir /container/a
5. echo $$ > /container/a/tasks (add tasks to the new container)
6. echo -n <num_pages> > /container/a/memory.limit
   example
   echo -n 204800 > /container/a/memory.limit, sets the limit to 800 MB
   on a system with 4K page size
7. run tasks, see the memory controller work
8. Report results, provide feedback
9. Develop/use new patches and go to step 1

Test Results

Results for version 3 of the patch were posted at
http://lwn.net/Articles/242554/

The code was also tested on a power box with regular machine usage scenarios,
the config disabled and with a stress suite that touched all the memory
in the system and was limited in a container.

Dhaval ran several tests on v6 and gave his thumbs up to the controller
(a hard to achieve goal :-)).

Run kernbench stress

Three simultaneously and with one inside a container of 800 MB.

Kernbench results running within the container of 800 MB

Thu Aug 16 22:34:59 IST 2007
2.6.23-rc2-mm2-mem-v6
Average Half load -j 4 Run (std deviation):
Elapsed Time 466.548 (47.6014)
User Time 876.598 (10.5273)
System Time 223.136 (1.29247)
Percent CPU 237.2 (23.2744)
Context Switches 146351 (6539.91)
Sleeps 174003 (5031.94)

Average Optimal load -j 32 Run (std deviation):
Elapsed Time 423.496 (60.625)
User Time 897.285 (23.0391)
System Time 228.836 (6.11205)
Percent CPU 257.1 (40.9022)
Context Switches 262134 (123397)
Sleeps 270815 (103597)

Kernbench results running within the default container

Thu Aug 16 22:34:33 IST 2007
2.6.23-rc2-mm2-mem-v6
Average Half load -j 4 Run (std deviation):
Elapsed Time 424.17 (3.45908)
User Time 841.992 (5.40178)
System Time 213.01 (0.706258)
Percent CPU 248.2 (0.83666)

Context Switches 134254 (9535.83)
Sleeps 167359 (6858.45)

Average Optimal load -j 32 Run (std deviation):
Elapsed Time 407.092 (108.932)
User Time 878.493 (38.6575)
System Time 222.155 (9.77127)
Percent CPU 278.4 (91.5826)
Context Switches 253836 (127708)
Sleeps 263760 (103468)

Kernbench results running within the default container

Thu Aug 16 22:34:52 IST 2007
2.6.23-rc2-mm2-mem-v6
Average Half load -j 4 Run (std deviation):
Elapsed Time 465.038 (48.5147)
User Time 874.742 (5.86563)
System Time 222.194 (0.561676)
Percent CPU 237.2 (22.5211)
Context Switches 144040 (7052.23)
Sleeps 172130 (5608.73)

Average Optimal load -j 32 Run (std deviation):
Elapsed Time 426.25 (62.34)
User Time 893.938 (20.6732)
System Time 227.717 (5.87502)
Percent CPU 255.6 (40.7163)
Context Switches 259560 (122953)
Sleeps 267402 (101801)

Ebizzy results
---

ebizzy on top of rc2-mm2 (without mem-controller) gave this

ebizzy 0.3, Copyright 2006 Intel Corporation
Written by Val Henson <val_henson@linux.intel.com
always_mmap 0
never_mmap 0
chunks 10
prevent coalescing using permissions 0
prevent coalescing using holes 0
records 1048576
records per thread 65536
random_size 0

chunk_size 262144
threads 16
verbose 1
linear 0
page size 4096
Allocated memory
Wrote memory
Threads starting
Threads finished

real 5m54.362s
user 33m49.707s
sys 4m11.100s

ebizzy with 2500 pages, and oprofiled.

ebizzy 0.3, Copyright 2006 Intel Corporation
Written by Val Henson <val_henson@linux.intel.com
always_mmap 0
never_mmap 0
chunks 10
prevent coalescing using permissions 0
prevent coalescing using holes 0
records 1048576
records per thread 65536
random_size 0
chunk_size 262144
threads 16
verbose 1
linear 0
page size 4096
Allocated memory
Wrote memory
Threads starting
Threads finished

real 6m15.561s
user 36m2.203s
sys 3m30.829s

Documentation

An article describing the design of the memory controller is available
at http://lwn.net/Articles/243795/

series

res_counters_infra.patch
mem-control-setup.patch
mem-control-accounting-setup.patch
mem-control-accounting.patch
mem-control-task-migration.patch
mem-control-lru-and-reclaim.patch
mem-control-out-of-memory.patch
mem-control-choose-rss-vs-rss-and-pagecache.patch
mem-control-per-container-page-referenced.patch

--
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