
Subject: [PATCH 0/8] RSS controller based on process containers (v3.1)

Posted by [Pavel Emelianov](#) on Mon, 04 Jun 2007 13:21:01 GMT

[View Forum Message](#) <> [Reply to Message](#)

Adds RSS accounting and control within a container.

Changes from v3

- comments across the code
- git-bisect safe split
- lost places to move the page between active/inactive lists

Ported above Paul's containers V10 with fixes from Balbir.

RSS container includes the per-container RSS accounting and reclamation, and out-of-memory killer.

Each mapped page has an owning container and is linked into its LRU lists just like in the global LRU ones. The owner of the page is the container that touched the page first. As long as the page stays mapped it holds the container, is accounted into its usage and lives in its LRU list. When page is unmapped for the last time it releases the container. The RSS usage is exactly the number of pages in its booth LRU lists, i.e. the number of pages used by this container. When this usage exceeds the limit set some pages are reclaimed from the owning container. In case no reclamation possible the OOM killer starts thinning out the container.

Thus the container behaves like a standalone machine - when it runs out of resources, it tries to reclaim some pages, and if it doesn't succeed, kills some task.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

The testing scenario may look like this:

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

And do other stuff. To check the reclamation to work we need a simple program that touches many pages of memory, like this:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>

#ifndef PGSIZE
#define PGSIZE 4096
#endif

int main(int argc, char **argv)
{
    unsigned long pages;
    int i;
    char *mem;

    if (argc < 2) {
        printf("Usage: %s <number_of_pages>\n", argv[0]);
        return 1;
    }

    pages = strtoul(argv[1], &mem, 10);
    if (*mem != '\0') {
        printf("Bad number %s\n", argv[1]);
        return 1;
    }

    mem = mmap(NULL, pages * PGSIZE, PROT_READ | PROT_WRITE,
               MAP_PRIVATE | MAP_ANON, 0, 0);
    if (mem == MAP_FAILED) {
        perror("map");
        return 2;
    }

    for (i = 0; i < pages; i++)
        mem[i * PGSIZE] = 0;

    printf("OK\n");
    return 0;
}
```
