Recent advances in the Linux kernel resource management

Kir Kolyshkin, OpenVZ
kir@openvz.org

Agenda

- Resources to account and control
- Some background on containers
- Existing functionality, shortcomings
- Control Groups a.k.a. cgroups
- Memory Controller
- Future work

Resources

Why?

- All resources are finite
- Multiple tasks and users
- Need usage statistics / bookkeeping
- Need Denial of Service protection
- Need Quality of Service level
  (not only limits but guarantees)

What?

- CPU
- Memory (RAM)
- Swap
- Disk space
- Disk I/O
- Network

Resources: CPU

CPU is given to tasks in time slices

- CPU shares/weights
- CPU limits
- for SMP: CPU affinity

Resources: Memory & swap

- User memory
  - Virtual and physical (RSS) memory
  - Dirty page cache
- Kernel memory
  - Various objects, different allocators
  - Special case: network buffers
- Swap space

Resources: disk

- Disk space
- Disk I/O bandwidth
  - read/write
  - mmap()
  - swapin/swapout
  - Problem: most of I/O is async

Resources: networking

- Network bandwidth: solved by tc
- **T**raffic **C**ontrol:
  - Shaping
  - Scheduling
  - Policies
  - Dropping

Containers

What are containers?

- Multiple isolated userspace instances
- Running on top of a single kernel
- Like VMs but very lightweight, native performance, low overhead

Containers Implementations

- OpenVZ
- Parallels Virtuozzo Containers
- FreeBSD jails
- Linux-VServer
- Solaris 10 Containers/Zones
- IBM AIX6 WPARs (Workload Partitions)

Containers cont'd

- Multiple containers should peacefully co-exist, need DoS protection
- From the resource management point of view, containers are just groups of processes.

Existing mechanisms

Disk Quota

- Per mount point disk quota for users and groups
- Soft limits, hard limits, grace periods
- Can see the current usage
- Can be inc'd/dec'd on-the-fly
- Applications are expecting disk space shortage (or at least should be)

CPU

- Per-process nice value which can be changed on-the-fly (`nice`, `renice`)
- Real-time priority queue
- Hard CPU time limit (`ulimit -c`)

ulimit

- setrlimit()/getrlimit() syscalls
- Controls 16 different resources:
  core file size, data seg size, scheduling priority, file size, pending signals,
  max locked memory, max memory size, number of open files, pipe size,
  POSIX message queues, real-time priority, stack size, cpu time, max user processes,
  virtual memory, file locks
- Soft limits and hard limits

ulimit: problems

- Not all resources are covered
- Ulimits set in the current context
  - the only good place to set is login
  - some can only be decreased run-time
- All limits are per-process
  - only NPROC is per-UID
- Current usage values are unknown
- Memory limits are mostly ignored

Control Groups

Control Groups

- A generic mechanism for grouping tasks into hierarchical groups
- Multiple resource controllers
- Possible to have different groups for different controllers
- Managed via cgroup filesystem

Control Groups: interface

Managed via cgroup filesystem:

```
mkdir /dev/cgroup
mount -t cgroup none /dev/cgroup
mkdir /dev/cgroup/0
cd /dev/cgroup/0
echo $$ > tasks
cat /proc/self/cgroup
/etc/init.d/httpd start
```

Control Groups: history

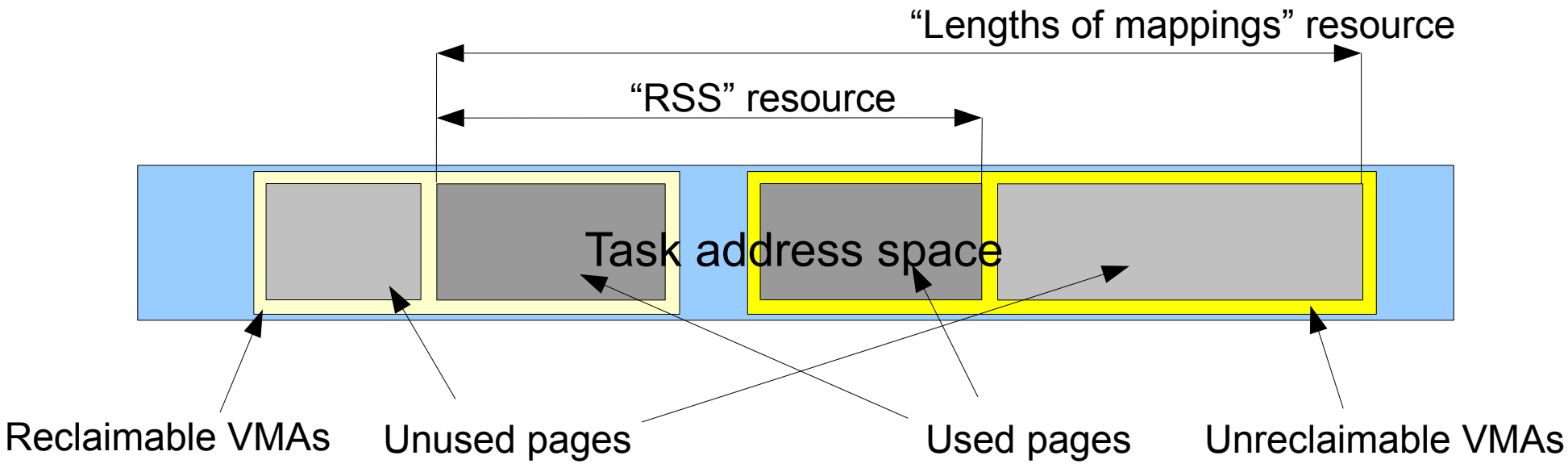
- A feature known as cpusets was developed by big iron Bull/SGI guys
- Used to maintain process groups to NUMA nodes affinity
- Paul Menage generalized it
- Now cpusets is just one of the resource controllers

Memory Controller

Memory controller

- User memory:
  - RSS
  - Page cache
- Reclamation
  - Same as `try_to_free_pages()`
- OOM killer

User Memory



VMAs classification

- *unreclaimable*:
  private and anonymous
- *reclaimable*:
  shared file mappings

Pages classification

- *unused*:
  parts of mapped regions
- *used*:
  touched pages

MemCtrl: interface

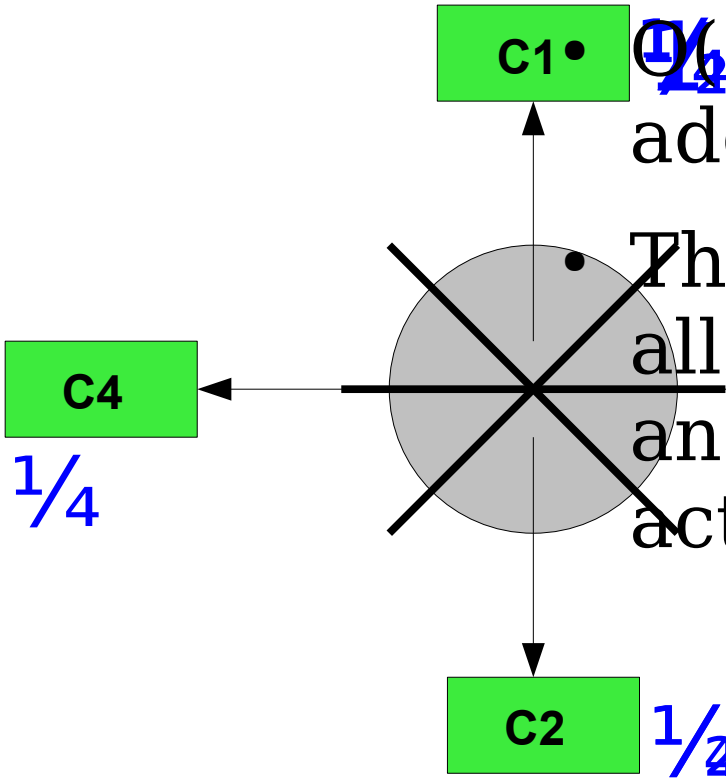
```
# echo 4M > memory.limit_in_bytes
# cat memory.limit_in_bytes
4194304
# cat memory.usage_in_bytes
172032
# cat memory.max_usage_in_bytes
294912
# cat memory.failcnt
0
# cat memory.stat
....
```

Shared Pages accounting

- Shared code/library segments
- Approaches:
  - Charge to the first user only (unfair)
  - Charge to all users (incorrect totals)
  - Charge a fraction to every user

Page fractions accounting

Algorithm benefits



- O(1) algorithm of adding and removing
- The sum of RSS on all beancounters is an amount of all actually used pages

Future

Future a.k.a. TODO

- Shared pages accounting
- VMA (user mappings) length ctrl
- Kernel memory controller
- cgroups checkpoint/restart
- per-cgroup I/O priorities
- All that is available in OpenVZ; needs to be ported to mainstream

More Info

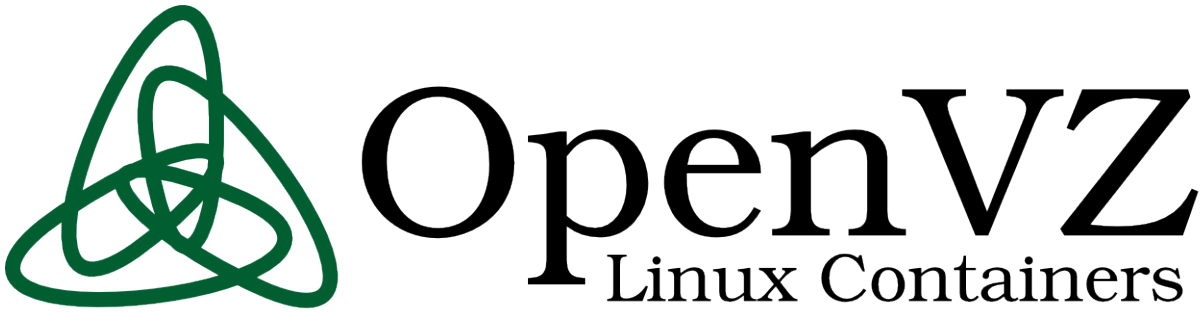
/usr/src/linux/Documentation/cgroups/*

/usr/src/linux/Documentation/controllers/*

containers@linux-foundation.org

Questions? Comments?

kir@openvz.org



Booth #63