Architect of an Open World™

Slurm Support for Linux Control Groups

Slurm User Group 2010, Paris, France, Oct 5th 2010

Martin Perry

Bull Information Systems

Phoenix, Arizona

martin.perry@bull.com

cgroups Concepts

- Control Groups (cgroups) are a mechanism for aggregating and partitioning processes (tasks) in Linux
- Introduced in kernel 2.6.24

<u>Concepts</u>

- **cgroup** – a group of tasks with shared characteristics
- **subsystem** – a module that applies parameters to cgroups to control them in particular ways, typically for resource management
- **hierarchy** – a set of cgroups organized in a hierarchical tree, plus one or more subsystems associated with that tree

Design is similar to Linux cpusets, but the subsystem concept provides a much larger set of controls

cgroups provide a form of lightweight virtualization

Slurm Support for cgroups – Why?

Why use cgroups in Slurm?

- To provide a **common framework** for implementing important Slurm features (process tracking, resource management, etc.)
- To improve **efficiency** of Slurm activities (e.g., process tracking, collection of accounting statistics)
- To improve **robustness** (e.g., more reliable process tracking, more reliable cleanup via release_agent mechanism)
- To **simplify** addition of new Slurm features (e.g., device resource management, network resource management)
- To make Slurm resource allocations visible outside Slurm

What are cgroups? - Subsystems

Current subsystems

- **cpuset** – controls access to individual CPUs and memory nodes by a cgroup
- **cpu** – schedules CPU access to cgroups
- **cpuacct** – reports CPU resource usage by a cgroup
- **memory** – controls access to memory resources and reports memory resource usage by a cgroup
- **devices** – controls access to devices by a cgroup; e.g., gpus
- **freezer** – suspends and resumes tasks in a cgroup
- **net_cls –** tags network packets in a cgroup to allow prioritizing of network traffic
- **oom** – controls the order in which tasks are killed in out-of-memory conditions
- **blkio** – tracks I/O ownership, allowing control of access to block I/O resources

What are cgroups? – Subsystem Parameters

Examples of subsystem parameters (state objects)

cpuset subsystem

cpuset.cpus: defines the set of cpus that the tasks in the cgroup are allowed to execute on

cpuset.mems: defines the set of memory zones that the tasks in the cgroup are allowed to use

memory subsystem

memory.limit_in_bytes: defines the memory limit for the tasks in the cgroup

memory.swappiness: controls kernel reclamation of memory from the tasks in the cgroup (swap priority)

freezer subsystem

freezer.state: controls whether tasks in the cgroup are active (runnable) or suspended

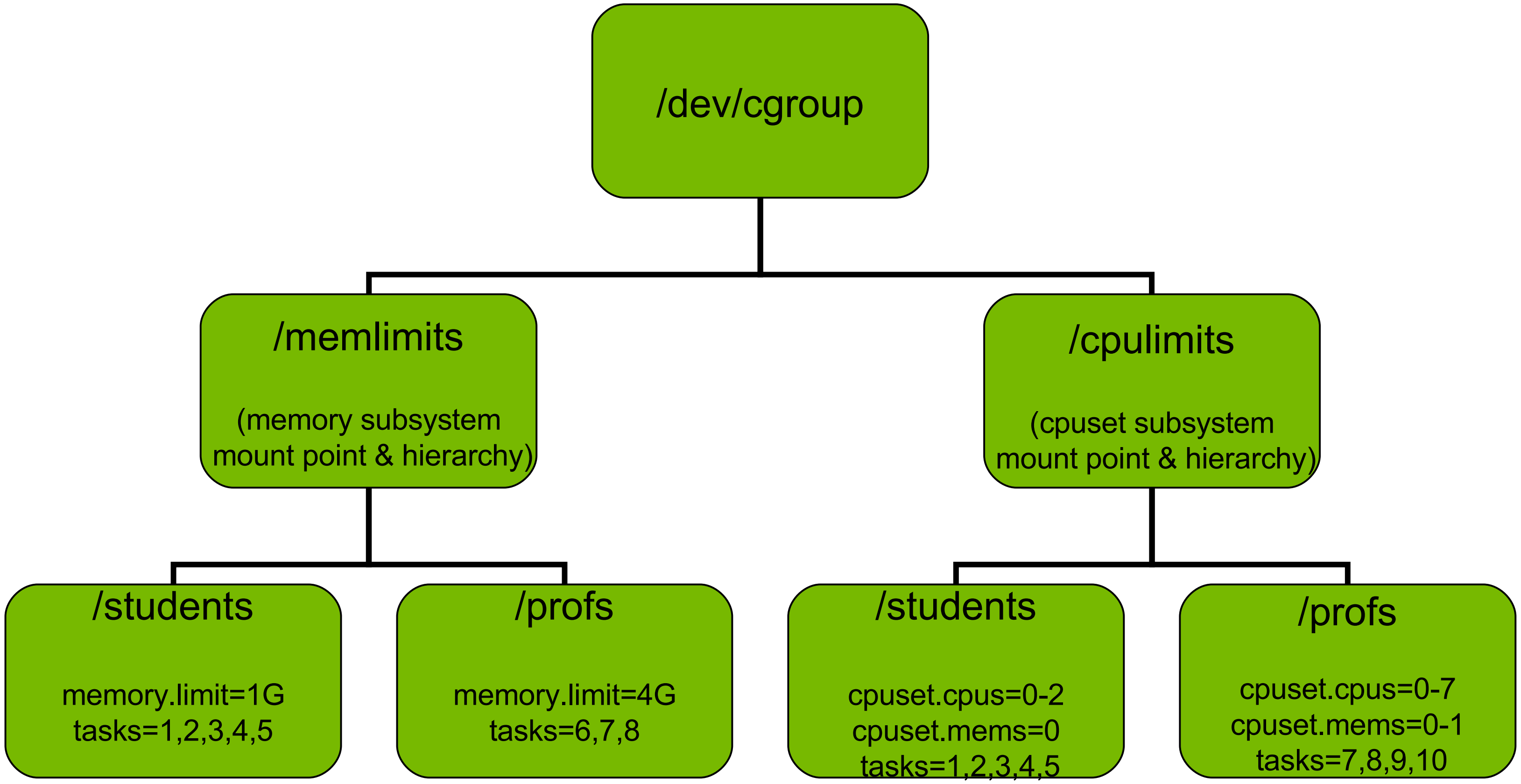
What are cgroups? – Important Features

Other important features of cgroups

- Represented as virtual file system under /dev/cgroup
- Users and applications interact with cgroups by manipulating directories and files in the cgroup virtual file system using standard shell commands and system calls (mkdir, mount, echo, read, etc.)
- Individual cgroups are represented as directories
- *tasks* file in each cgroup directory lists the tasks (pids) in that cgroup
- New tasks are automatically added to the cgroup of their parent
- Tasks are automatically removed from a cgroup when they terminate or are added to a different cgroup in the same hierarchy
- Hierarchies are created by mounting subsystems, using the *mount* command; subsystem names are specified as mount options
- Subsystem parameters (state objects) are represented as a set of files in each cgroup in a hierarchy, with values that apply only to that cgroup
- Each task is present in only one cgroup in each hierarchy
- Includes mechanism for automatic removal of abandoned cgroups (release_agent)

What are cgroups? - Example

The following figure depicts a simple cgroup scenario for a university computer system that requires different sets of memory and cpu resource limits for students and professors



What are cgroups? - Summary

<u>Summary</u>

cgroups provide a flexible, dynamic and efficient mechanism for controlling and querying groups of tasks

<u>Important applications</u>

- Resource management (e.g., cpu, memory, device constraints)
- Process tracking & suspend/resume (used by Slurm)
- Collecting resource usage statistics (e.g., memory usage stats from memory subsystem, cpu usage stats from cpuacct subsystem)

Slurm Support for cgroups – Slurm Features

Slurm features implemented with cgroups

- Process tracking
  - New proctrack/cgroup plugin
  - Alternative to existing proctrack/linuxproc plugin
  - Configured with ProctrackType=proctrack/cgroup in slurm.conf
  - Uses freezer subsystem to track processes and suspend/resume job steps
  - Transparent to users: No changes to user command formats or options

- Resource constraints
  - New task/cgroup plugin
  - Alternative to existing task/affinity plugin
  - Configured with TaskPlugin=task/cgroup in slurm.conf
  - Currently supports only cpu binding to tasks (task affinity), using the cpuset subsystem
  - Transparent to users: No changes to user command formats or options
  - We plan to add support for additional subsystems in the future to allow constraints on other types of resources (memory, devices, etc.)

Slurm Support for cgroups – Cgroup Organization

Organization of Slurm cgroups

- Master slurm cgroup directory at /dev/cgroup/slurm
- Separate hierarchy created for each required subsystem, e.g.

  /dev/cgroup/slurm/freezer
  /dev/cgroup/slurm/cpuset

  A slurm cgroup hierarchy is also called a *namespace*

- Within each namespace, cgroup structures are created dynamically as needed to represent Slurm users, jobs, steps and tasks, in the following format:

  /dev/cgroup/slurm/<namespace>/uid_%uid/job_%jobid/step_%stepid/task_%taskid

  Depending on the slurm features a namespace is used to implement, only part of this structure may be required.

Slurm Support for cgroups - API

Slurm cgroup API

- Set of data and methods in src/common for creating, modifying, querying and deleting Slurm cgroups
- Provides a common interface to all Slurm components for using cgroups
- Hides some implementation details

Primary API functions

- Create/delete a namespace
- Mount/unmount a namespace (subsystem)
- Create/delete a cgroup (within a namespace)
- Add a task (pid) to a cgroup
- Get the list of tasks in a cgroup
- Set/get the value of a cgroup parameter

Slurm Support for cgroups – cgroup.conf

Slurm cgroup configuration file – cgroup.conf

- Resides in same directory as slurm.conf
- Read by all Slurm components that use cgroups (currently only proctrack/cgroup and task/cgroup plugins)
- Defines general configuration parameters for Slurm cgroups, e.g.

  CgroupSubsystems – list of subsystems (namespaces) to be created/mounted by task/cgroup plugin (currently supports only cpuset subsystem)

  CgroupReleaseAgent – specifies location of release_agent files to be run when a cgroup is abandoned
- Also defines feature-specific cgroup parameters, e.g.

  TaskAffinityBindType - defines cpu binding type when TaskPlugin=task/cgroup is configured

Slurm Support for cgroups - Example

```
[sulu] (slurm) etc> cat slurm.conf | grep cgroup
ProctrackType=proctrack/cgroup
TaskPlugin=task/cgroup
[sulu] (slurm) etc> cat cgroup.conf | grep TaskAffinity
TaskAffinityBindType=“threads"
[sulu] (slurm) etc> srun -n2 sleep 500 &
[1] 8639
[sulu] (slurm) etc> squeue
JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
634   phoenix    sleep    slurm   R       0:04      1 n13
[sulu] (slurm) etc> scontrol listpids
PID      JOBID    STEPID LOCALID GLOBALID
8662     634      0      0       0
8663     634      0      1       1
[sulu] (slurm) etc> cat /dev/cgroup/slurm/freezer/uid_200/job_634/step_0/tasks
8662
8663
[sulu] (slurm) etc> cat /dev/cgroup/slurm/freezer/uid_200/job_634/step_0/freezer.state
THAWED
[sulu] (slurm) etc> scontrol suspend 634
[sulu] (slurm) etc> cat /dev/cgroup/slurm/freezer/uid_200/job_634/step_0/freezer.state
FROZEN
[sulu] (slurm) etc> squeue
JOBID PARTITION   NAME     USER  ST       TIME  NODES NODELIST(REASON)
634   phoenix    sleep    slurm   S       0:45      1 n13
[sulu] (slurm) etc> scontrol resume 634
[sulu] (slurm) etc> cat /dev/cgroup/slurm/freezer/uid_200/job_634/step_0/freezer.state
THAWED
[sulu] (slurm) etc> squeue
JOBID PARTITION   NAME     USER  ST       TIME  NODES NODELIST(REASON)
634   phoenix    sleep    slurm   R       0:48      1 n13
```

© Bull, 2010

Slurm Support for cgroups – Example (contd)

```
[sulu] (slurm) etc> srun -n2 sleep 500 &
[1] 8639

0
0
0

[sulu] (slurm) etc> cat /dev/cgroup/slurm/cpuset/uid_200/job_634/step_0/task_0/tasks
8662
[sulu] (slurm) etc> cat /dev/cgroup/slurm/cpuset/uid_200/job_634/step_0/task_0/cpuset.cpus
7
[sulu] (slurm) etc> cat /dev/cgroup/slurm/cpuset/uid_200/job_634/step_0/task_1/tasks
8663
[sulu] (slurm) etc> cat /dev/cgroup/slurm/cpuset/uid_200/job_634/step_0/task_1/cpuset.cpus
15
[sulu] (slurm) etc> cat /proc/8662/status | grep Cpus_allowed_list
Cpus_allowed_list:      7
[sulu] (slurm) etc> cat /proc/8663/status | grep Cpus_allowed_list
Cpus_allowed_list:      15
```

© Bull, 2010

Slurm Support for cgroups – History & Status

Development History & Status

- Primary design and development by Matthieu Hautreux of CEA

  - Original patch from Matthieu supported process tracking and memory resource limits using memory subsystem

  - Second patch from Matthieu replaced memory subsystem with freezer subsystem due to memory subsystem performance problems with large numbers of cores. Included in Slurm version 2.2.0-pre2

- Expanded and enhanced by Martin Perry of Bull (Yiannis Georgiou, Rod Schultz) in consultation with Matthieu and LLNL

  - Second patch from Matthieu refactored to provide more general support of cgroups

  - Added task/cgroup plugin and support for task containment using the cpuset subsystem

  - Expected to be included in Slurm version 2.2, or a pre-release of version 2.3

Slurm Support for cgroups – Future Development

Future Development

- Simplification/consolidation of code involved in cpu selection/distribution for task binding to cpus (lllp_distribution)
- Support for constraints on interactive user shells through Pluggable Authentication Module (PAM) by restricting user-level access to resources with user cgroups (uid_%uid)
- Support for memory, device and blkio subsystems to apply constraints on those resource types to Slurm users, jobs, steps or tasks
- Support for memory and cpuacct subsystems to collect accounting statistics (jobacct_gather plugin)

Questions?

Architect of an Open World™

LIBERATE IT