Tutorial 3: Cgroups Support On SLURM

SLURM User Group 2012, Barcelona, October 9-10th 2012

Martin Perry
email: martin.perry@bull.com
Yiannis Georgiou
email: yiannis.georgiou@bull.fr
Matthieu Hautreux
email: matthieu.hautreu@cea.fr

Outline

- Introduction to cgroups
- Cgroups implementation in SLURM
  - Basic API for cgroups usage
  - Organization and Configuration Issues
- Cgroups subsystems support for SLURM
  - Stable: freezer, cpuset, memory, devices
  - Experimental: cpuacct
  - Planned: blkio, net_cls
- Summary and Perspectives

Introduction to Cgroups

Control Groups (cgroups) is a **Linux kernel mechanism** (appeared in 2.6.24) to limit, isolate and monitor resource usage (CPU, memory, disk I/O, etc.) of groups of processes.

Features

- ***Resource Limiting*** (i.e. not to exceed a memory limit)
- ***Prioritization*** (i.e. groups may have larger share of CPU)
- ***Isolation*** (i.e. isolate GPUs for particular processes)
- ***Accounting*** (i.e. monitor resource usage for processes)
- ***Control*** (i.e. suspending and resuming processes)

Cgroups Model and Concepts

Model

Cgroups **similar** to Linux processes:

- Hiererachical
- Inheritance of attributes from parent to child

but **different** because:

- **multiple hiererachies** of cgroups may exist that are attached to one or more subsystems

Concepts

- **Cgroup** – a group of processes with the same characteristics
- **Hierarchy** – a set of cgroups organized in a tree, plus one or more subsystems associated with that tree
- **Subsystem** – a module that applies parameters to a group of processes (cgroup)

Cgroups subsystems

- **cpuset** – assigns tasks to individual CPUs and memory nodes in a cgroup
- **cpu** – schedules CPU access to cgroups
- **cpuacct** – reports CPU resource usage of tasks of a cgroup
- **memory** – set limits on memory use and reports memory usage for a cgroup
- **devices** – allows or denies access to devices (i.e. gpus) for tasks of a cgroup
- **freezer** – suspends and resumes tasks in a cgroup
- **net_cls** – tags network packets in a cgroup to allow network traffic priorities
- **ns** – namespace subsystem
- **blkio** – tracks I/O ownership, allowing control of access to block I/O resources

Cgroups functionality rules

- Cgroups are represented as **virtual file systems**
  - Hierarchies are directories, created by mounting subsystems, using the mount command; subsystem names specified as mount options
  - Subsystem parameters are represented as files in each hierarchy with values that apply only to that cgroup
- **Interaction with cgroups** take place by manipulating directories and files in the cgroup virtual file system using standard shell commands and system calls (mkdir, mount, echo, etc)
  - *tasks* file in each cgroup directory lists the tasks (pids) in that cgroup
  - Tasks are automatically removed from a cgroup when they terminate or are added to a different cgroup in the same hierarchy
  - Each task is present in only one cgroup in each hierarchy
- Cgroups have a mechanism for **automatic removal** of abandoned cgroups (release_agent)

Cgroups subsystems parameters

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)

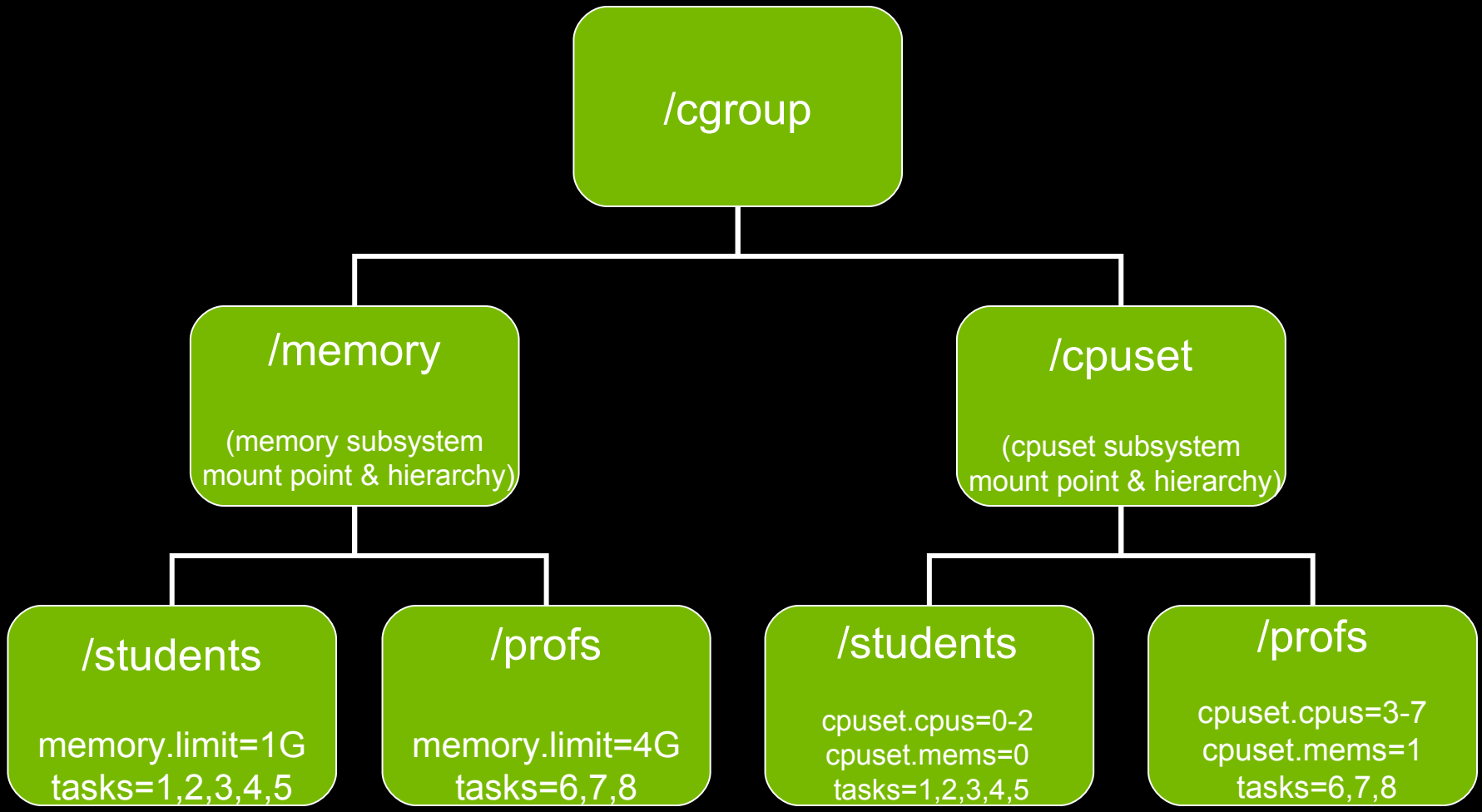
freezer subsystem

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

devices subsystem

devices_allow: specifies devices to which tasks in a cgroup have access

Cgroups functionality example



Cgroups Functionality Example

```
[root@mordor:~]# mkdir /cgroup
[root@mordor:~]# mkdir /cgroup/cpuset
[root@mordor:~]# mount -t cgroup -o cpuset none /cgroup/cpuset
[root@mordor:~]# ls /cgroup/cpuset/
cpuset.cpus  cpuset.mems tasks notify_on_release release_agent
[root@mordor:~]# mkdir /cgroup/cpuset/students
[root@mordor:~]# mkdir /cgroup/cpuset/profs
[root@mordor:~]# /bin/echo 0-2 > /cgroup/cpuset/students/cpuset.cpus
[root@mordor:~]# /bin/echo 0 > /cgroup/cpuset/students/cpuset.mems
[root@mordor:~]# /bin/echo $PIDS_students > /cgroup/cpuset/students/tasks
[root@mordor:~]# /bin/echo 3-7 > /cgroup/cpuset/profs/cpuset.cpus
[root@mordor:~]# /bin/echo 1 > /cgroup/cpuset/profs/cpuset.mems
[root@mordor:~]# /bin/echo $PIDS_profs > /cgroup/cpuset/profs/tasks
```

Why Support Cgroups In SLURM?

- To improve **tasks isolation** upon resources
- To provide a **common framework** for resources isolation, limitations and usage accounting
- To improve **efficiency** of SLURM activities (e.g., process tracking, collection of accounting statistics)
- To improve **robustness** (e.g. more reliable cleanup via release_agent mechanism)
- To simplify the addition of **new features** like management of network bandwidth or disks I/O as individual resources

Cgroups Implementation in SLURM

- A common API to manage cgroup directories and files
  - src/common/xcgroup.{h,c}
  - src/common/xcgroup_read_config.{h,c}
- Three plugins that add cgroup related features to slurmd
  - **proctrack/cgroup**: track/suspend/resume job's tasks
  - **task/cgroup**: confine tasks to the allocated resources
  - **jobacct_gather/cgroup**: collect accounting statistics
- A dedicated cgroup release_agent
  - Lock/Unlock cgroup hierarchy when managing slurm related cgroups to avoid race conditions

SLURM Cgroups API

Ease cgroup init, directories and files management

- slurm_cgroup_conf_t
  - Stores cgroup related conf
  -
- xcgroup_ns_t
  - Structure associated with a cgroup hierarchy
  - Helps to initialize/mount/umount/search_into it
- xcgroup_t
  - Structure associated to a cgroup directory
  - Linked to the associated xcgroup_ns
  - Helps to add/get tasks, set/get params
  - Helps to lock/unlock the underlying directory

Organization of SLURM Cgroups

All SLURM cgroups use a common format in the virtual file system. Base directory is /cgroup (default) or as configured in cgroup.conf configuration file:

/cgroup/%subsystem/slurm/uid_%uid/job_%jobid/step_%stepid/task_%taskid

This structure is specific to each compute node on which a job/step/task has been allocated resources. Jobs and steps that use multiple nodes will have a cgroup structure on each node

Configuration (slurm.conf & cgroup.conf)

```
[root@leaf ~]# grep cgroup /etc/slurm/slurm.conf
ProctrackType=proctrack/cgroup
TaskPlugin=task/cgroup
[root@leaf ~]#
[root@leaf ~]# cat /etc/slurm/cgroup.conf
###
#
# Slurm cgroup support configuration file
#
# See man slurm.conf and man cgroup.conf for further
# information on cgroup configuration parameters
#--
CgroupAutomount=yes
CgroupReleaseAgentDir="/etc/slurm/cgroup"

ConstrainCores=yes
TaskAffinity=yes
[root@leaf ~]#
```

SLURM Cgroups Documentation

- **Cgroups Guide**
  (www.schedmd.com/slurmdocs/cgroups.html)

- **slurm.conf man page**
  ProctrackType=proctrack/cgroup
  TaskPlugin=task/cgroup
  JobacctGatherType=jobacct_gather/cgroup

- **cgroup.conf man page**
  common cgroup options + options specific to each plugin

proctrack/cgroup plugin: **freezer** subsystem

Track jobs processes using the freezer subsystem

- Every spawned process is tracked
  - Automatic inheritance of parent's cgroup
  - No way to escape the container
- Every processes can be frozen
  - Using the Thawed | Frozen state of the subsystem
  - No way to avoid the freeze action

proctrack/cgroup plugin: **freezer** subsystem

[sulu] (slurm) mnp> srun -p sulu-only sleep 5000 &

```
[sulu] (slurm) mnp> cat /cgroup/freezer/slurm/uid_200/job_259/step_0/tasks
2350
[sulu] (slurm) mnp> cat /cgroup/freezer/slurm/uid_200/job_259/step_0/freezer.state
THAWED
[sulu] (slurm) mnp> scontrol suspend 259
[sulu] (slurm) mnp> cat /cgroup/freezer/slurm/uid_200/job_259/step_0/freezer.state
FROZEN
[sulu] (slurm) mnp> ps -ef f | tail -n 2
root     2339     1  0 14:51 ?        Sl     0:00 slurmstepd: [259.0]
slurm    2350  2339  0 14:51 ?        T      0:00  \_ /bin/sleep 5000
[sulu] (slurm) mnp> scontrol resume 259
[sulu] (slurm) mnp> cat /cgroup/freezer/slurm/uid_200/job_259/step_0/freezer.state
THAWED
[sulu] (slurm) mnp> ps -ef f | tail -n 2
root     2339     1  0 14:51 ?        Sl     0:00 slurmstepd: [259.0]
slurm    2350  2339  0 14:51 ?        S      0:00  \_ /bin/sleep 5000
```

task/cgroup plugin

Constrain jobs tasks to the allocated resources

- 3 independent layers of managed resources using 3 subsystems
  - Cores/CPUs (**cpuset**), Memory (**memory**), GRES (**devices**)
- Every spawned process is tracked
  - Automatic inheritance of parent's cgroup
  - No way to use additional resources
- Each layer has its own additional parameters
- More resources could be added

task/cgroup plugin : **cpuset** subsystem

Constrain jobs tasks to the allocated cores

- Configurable feature
  - ConstrainCores=yes|no
- Use step's allocated cores with “exclusive steps”
  - Otherwise, let steps use job's allocated cores
- Basic affinity management as a configurable sub-feature
  - TaskAffinity=yes|no in cgroup.conf (rely on HWLOC)
  - Supports block and cyclic distribution of allocated CPUs to tasks for affinity

task/cgroup plugin : **cpuset** subsystem

TaskAffinity binding logic

- Detect the number of allocated cores
- Look at the requested binding (--cpu_bind)
- Use the granularity that best matches the allocated resources versus the number of tasks to spawn * the req cores per task
  - If sockets are requested but less sockets than tasks, automatically switch to cores (1)
  - If cores are requested but less cores than required, automatically switch to PU (hyperthread) (2)
  - If less PU than required, disable affinity

task/cgroup plugin : **cpuset** subsystem

TaskAffinity binding logic (following)

- Distribute the associated objects to tasks (socket|core(|PU))
- Relax constraint to match the requested binding if necessary
  - In (1), each task is then allowed to access other cores sharing the partial sockets already allowed
  - In (2), each task is then allowed to access the other hyperthreads sharing the partial cores already allowed

task/cgroup plugin : **cpuset** subsystem

TaskAffinity binding common behavior

- Distribute allocated cores to tasks for binding using either block or cyclic distribution.
- If cpu_bind > cores (socket, ldom)
  - Allow adjacent cores on already allocated object

task/cgroup plugin : **cpuset** subsystem

```
[sulu] (slurm) mnp> salloc -p chekov-only --exclusive srun -n1 --cpu_bind=none sleep 3000
salloc: Granted job allocation 260
```

```
[sulu] (slurm) etc> egrep "Cores|Affinity" cgroup.conf
ConstrainCores=yes
TaskAffinity=yes

[chekov] (slurm) mnp> cat /var/tmp/mnp-slurm/slurmd.log.chekov | grep task/cgroup
[2012-09-18T11:15:57] debug:  task/cgroup: now constraining jobs allocated cores
[2012-09-18T11:15:57] task/cgroup: loaded
[2012-09-18T11:16:56] [260.0] task/cgroup: now constraining jobs allocated cores
[2012-09-18T11:16:56] [260.0] task/cgroup: loaded
[2012-09-18T11:16:56] [260.0] task/cgroup: job abstract cores are '0-7'
[2012-09-18T11:16:56] [260.0] task/cgroup: step abstract cores are '0-7'
[2012-09-18T11:16:56] [260.0] task/cgroup: job physical cores are '0-7'
[2012-09-18T11:16:56] [260.0] task/cgroup: step physical cores are '0-7'
[2012-09-18T11:16:56] [260.0] task/cgroup: task[0] is requesting no affinity
```

task/cgroup plugin : **cpuset** subsystem

```
[sulu] (slurm) mnp> salloc -p chekov-only --exclusive srun -n1 --exclusive --cpu_bind=none
    sleep 3000
salloc: Granted job allocation 261
```

```
[chekov] (slurm) mnp>cat /var/tmp/mnp-slurm/slurmd.log.chekov | grep task/cgroup
[2012-09-18T13:43:16] [261.0] task/cgroup: now constraining jobs allocated cores
[2012-09-18T13:43:16] [261.0] task/cgroup: now constraining jobs allocated memory
[2012-09-18T13:43:16] [261.0] task/cgroup: loaded
[2012-09-18T13:43:16] [261.0] task/cgroup: job abstract cores are '0-7'
[2012-09-18T13:43:16] [261.0] task/cgroup: step abstract cores are '0'
[2012-09-18T13:43:16] [261.0] task/cgroup: job physical cores are '0-7'
[2012-09-18T13:43:16] [261.0] task/cgroup: step physical cores are '0'
[2012-09-18T13:43:16] [261.0] task/cgroup: task[0] is requesting no affinity
```

task/cgroup plugin : **cpuset** subsystem

```
[sulu] (slurm) mnp> salloc --exclusive srun -n2 --cpu_bind=cores sleep 3000
salloc: Granted job allocation 264
```

```
[chekov] (slurm) mnp> cat /var/tmp/mnp-slurm/slurmd.log.chekov | grep task/cgroup
[2012-09-18T14:13:08] [264.0] task/cgroup: now constraining jobs allocated cores
[2012-09-18T14:13:08] [264.0] task/cgroup: loaded
[2012-09-18T14:13:08] [264.0] task/cgroup: job abstract cores are '0-7'
[2012-09-18T14:13:08] [264.0] task/cgroup: step abstract cores are '0-7'
[2012-09-18T14:13:08] [264.0] task/cgroup: job physical cores are '0-7'
[2012-09-18T14:13:08] [264.0] task/cgroup: step physical cores are '0-7'
[2012-09-18T14:13:08] [264.0] task/cgroup: task[1] is requesting core level binding
[2012-09-18T14:13:08] [264.0] task/cgroup: task[0] is requesting core level binding
[2012-09-18T14:13:08] [264.0] task/cgroup: task[1] using Core granularity
[2012-09-18T14:13:08] [264.0] task/cgroup: task[0] using Core granularity
[2012-09-18T14:13:08] [264.0] task/cgroup: task[1] using cyclic distribution, task_dist 2
[2012-09-18T14:13:08] [264.0] task/cgroup: task[0] using cyclic distribution, task_dist 2
[2012-09-18T14:13:08] [264.0] task/cgroup: task[1] taskset '0x00000002' is set
[2012-09-18T14:13:08] [264.0] task/cgroup: task[0] taskset '0x00000001' is set
```

task/cgroup plugin : **cpuset** subsystem

```
[sulu] (slurm) mnp> salloc -p chekov-only  --exclusive srun -n1 --cpu_bind=socket sleep 3000
salloc: Granted job allocation 267
```

```
[chekov] (slurm) mnp> cat /var/tmp/mnp-slurm/slurmd.log.chekov | grep task/cgroup
[2012-09-18T14:23:42] [267.0] task/cgroup: now constraining jobs allocated cores
[2012-09-18T14:23:42] [267.0] task/cgroup: loaded
[2012-09-18T14:23:42] [267.0] task/cgroup: job abstract cores are '0-7'
[2012-09-18T14:23:42] [267.0] task/cgroup: step abstract cores are '0-7'
[2012-09-18T14:23:42] [267.0] task/cgroup: job physical cores are '0-7'
[2012-09-18T14:23:42] [267.0] task/cgroup: step physical cores are '0-7'
[2012-09-18T14:23:42] [267.0] task/cgroup: task[0] is requesting socket level binding
[2012-09-18T14:23:42] [267.0] task/cgroup: task[0] using Socket granularity
[2012-09-18T14:23:42] [267.0] task/cgroup: task[0] using cyclic distribution, task_dist 1
[2012-09-18T14:23:42] [267.0] task/cgroup: task[0] taskset '0x00000055' is set
```

task/cgroup plugin : **cpuset** subsystem

```
[sulu] (slurm) mnp> salloc -p chekov-only --exclusive srun -n1 --cpu_bind=ldom sleep 3000 &
salloc: Granted job allocation 268
```

```
[chekov] (slurm) mnp> cat /var/tmp/mnp-slurm/slurmd.log.chekov | grep task/cgroup
[2012-09-18T14:32:10] [268.0] task/cgroup: now constraining jobs allocated cores
[2012-09-18T14:32:10] [268.0] task/cgroup: loaded
[2012-09-18T14:32:10] [268.0] task/cgroup: job abstract cores are '0-7'
[2012-09-18T14:32:10] [268.0] task/cgroup: step abstract cores are '0-7'
[2012-09-18T14:32:10] [268.0] task/cgroup: job physical cores are '0-7'
[2012-09-18T14:32:10] [268.0] task/cgroup: step physical cores are '0-7'
[2012-09-18T14:32:10] [268.0] task/cgroup: task[0] is requesting ldom level binding
[2012-09-18T14:32:10] [268.0] task/cgroup: task[0] using Core granularity
[2012-09-18T14:32:10] [268.0] task/cgroup: task[0] using cyclic distribution, task_dist 1
[2012-09-18T14:32:10] [268.0] task/cgroup: task[0] higher level Machine found
[2012-09-18T14:32:10] [268.0] task/cgroup: task[0] taskset '0x000000ff' is set
```

task/cgroup plugin : **cpuset** subsystem

```
[sulu] (slurm) mnp> salloc -p chekov-only --exclusive srun -n2 --cpu_bind=socket sleep 3000
salloc: Granted job allocation 335
```

```
[chekov] (slurm) mnp> cat /var/tmp/mnp-slurm/slurmd.log.chekov | grep task/cgroup
[2012-09-27T03:28:06] [335.0] task/cgroup: now constraining jobs allocated cores
[2012-09-27T03:28:06] [335.0] task/cgroup: loaded
[2012-09-27T03:28:06] [335.0] task/cgroup: job abstract cores are '0-7'
[2012-09-27T03:28:06] [335.0] task/cgroup: step abstract cores are '0-7'
[2012-09-27T03:28:06] [335.0] task/cgroup: job physical cores are '0-7'
[2012-09-27T03:28:06] [335.0] task/cgroup: step physical cores are '0-7'
[2012-09-27T03:28:06] [335.0] task/cgroup: task[1] is requesting socket level binding
[2012-09-27T03:28:06] [335.0] task/cgroup: task[1] using Socket granularity
[2012-09-27T03:28:06] [335.0] task/cgroup: task[1] using cyclic distribution, task_dist 2
[2012-09-27T03:28:06] [335.0] task/cgroup: task[1] taskset '0x000000aa' is set
[2012-09-27T03:28:06] [335.0] task/cgroup: task[0] is requesting socket level binding
[2012-09-27T03;28:06] [335.0] task/cgroup: task[0] using Socket granularity
[2012-09-27T03:28:06] [335.0] task/cgroup: task[0] using cyclic distribution, task_dist 2
[2012-09-27T03:28:06] [335.0] task/cgroup: task[0] taskset '0x00000055' is set
```

task/cgroup plugin : **memory** subsystem

Constrain jobs tasks to the allocated amount of memory

- Configurable feature
  - ConstrainRAMSpace=yes|no
  - ConstrainSwapSpace=yes|no
- Use step's allocated amount of memory with “exclusive steps”
  - Otherwise, let steps use job's allocated amount
- Both RSS and swap can be monitored
- Trigger OOM killer on the cgroup's tasks when reaching limits
- Tolerant mechanism
  - AllowedRAMSpace , AllowedSwapSpace percents

task/cgroup plugin : **memory** subsystem

```
[mat@slacklap slurm]$ salloc --exclusive --mem-per-cpu 100 srun -n1 sleep 3000
salloc: Granted job allocation 67
```

```
[root@slacklap ~]# tail -f /var/log/slurmd.leaf10.log |grep task/cgroup
[2011-09-16T17:55:20] [67.0] task/cgroup: now constraining jobs allocated memory
[2011-09-16T17:55:20] [67.0] task/cgroup: loaded
[2011-09-16T17:55:20] [67.0] task/cgroup: job mem.limit=3520MB memsw.limit=3840MB
[2011-09-16T17:55:20] [67.0] task/cgroup: step mem.limit=3520MB memsw.limit=3840MB
```

```
[mat@slacklap slurm]$ salloc --exclusive --mem-per-cpu 100 srun –exclusive -n1
    sleep 3000
```

```
[root@slacklap ~]# tail -f /var/log/slurmd.leaf10.log |grep task/cgroup
[2011-09-16T17:57:31] [68.0] task/cgroup: now constraining jobs allocated memory
[2011-09-16T17:57:31] [68.0] task/cgroup: loaded
[2011-09-16T17:57:31] [68.0] task/cgroup: job mem.limit=3520MB memsw.limit=3840MB
[2011-09-16T17:57:31] [68.0] task/cgroup: step mem.limit=110MB memsw.limit=120MB
```

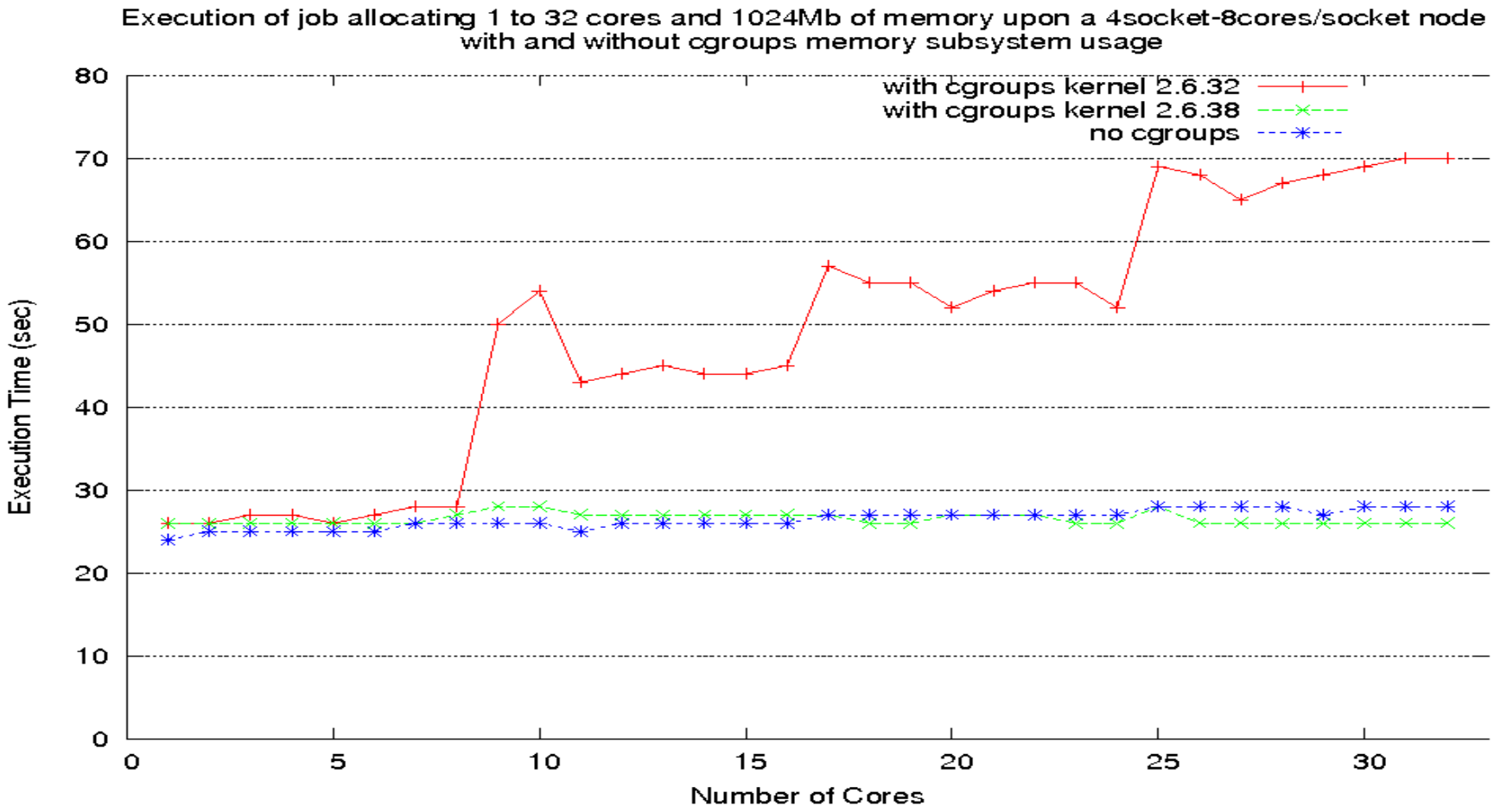
task/cgroup plugin : **memory** subsystem

Limitations

- Automatic cleaning of cgroup directories
  - when last byte is unattached
  - Can take a long long long time
- Performances penalities on some systems
  - Depending on the kernel/cgroup version
  - Depending on the NUMA architecture of the nodes

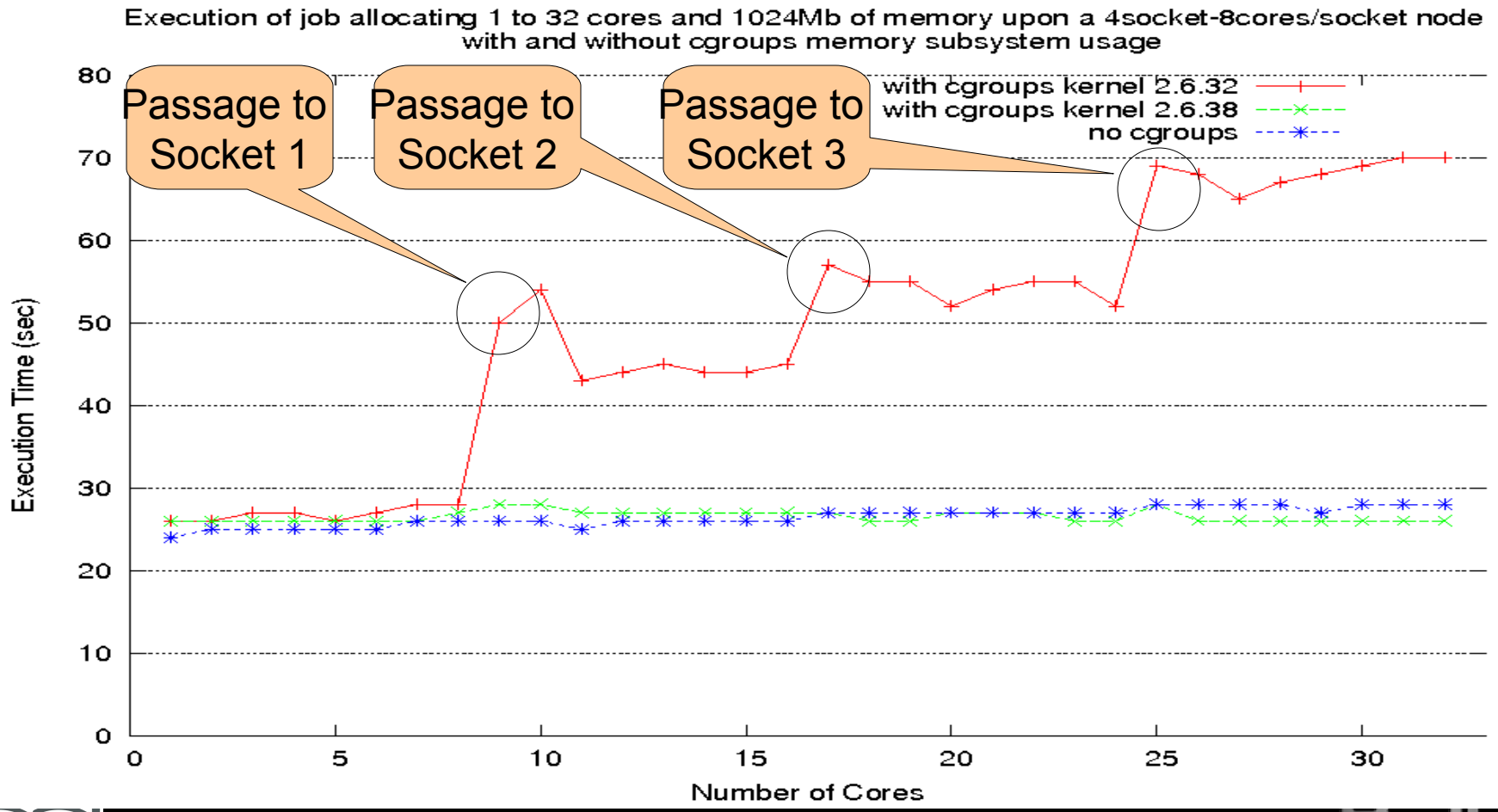
task/cgroup plugin : **memory** subsystem Problems Limitations

Performance degradation issues with cgroups memory and 2.6.32 kernel on 4socket-8core/socket machines



task/cgroup plugin : **memory** subsystem Problems Limitations

Performance degradation issues with cgroups memory and 2.6.32 kernel on 4socket-8core/socket machines



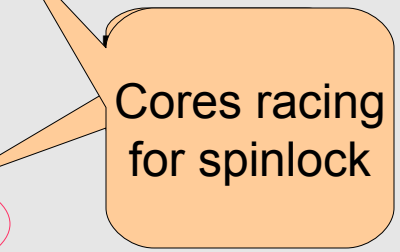
task/cgroup plugin : **memory** subsystem Problems Limitations

PerfTop with kernel 2.6.32 and 4socket-8cores/socket

Problem reported to cgroups Maintainers

PerfTop: 31987 irqs/sec kernel:80.2% exact: 0.0% 1000Hzcycles], (all, 32 CPUs)

samples	pcnt	function	DSO
156990.00	74.3%	_spin_lock	[kernel.kallsyms]
41694.00	19.7%	main	/tmp/memtests/malloc
3641.00	1.7%	clear_page_c	[kernel.kallsyms]
2558.00	1.2%	res_counter_charge	[kernel.kallsyms]
1750.00	0.8%	__alloc_pages_nodemask	[kernel.kallsyms]
1717.00	0.8%	__mem_cgroup_commit_charge	[kernel.kallsyms]



task/cgroup plugin : **memory** subsystem Improvements

PerfTop with kernel 2.6.38 and 4socket-8cores/socket

Problem corrected by cgroups maintainers

PerfTop: 31144 irqs/sec kernel:0.6% exact: 0.0% 1000Hzcycles], (all, 32 CPUs)

samples	pcnt	function	DSO
352809.00	97.5%	main	/tmp/memtests/malloc
2982.00	0.8%	clear_page_c	[kernel.kallsyms]
1019.00	0.3%	__alloc_pages_nodemask	[kernel.kallsyms]
725.00	0.2%	page_fault	[kernel.kallsyms]
279.00	0.1%	ktime_get	[kernel.kallsyms]
203.00	0.1%	get_page_from_freelist	[kernel.kallsyms]
165.00	0.0%	do_raw_spin_lock	[kernel.kallsyms]

Ticket spinlock Optimized performance

task/cgroup plugin : **devices** subsystem

Constrain jobs tasks to the allocated system devices

- Based on the **GRES** allocated resources and built upon the cgroup task plugin
  - Each task is allowed to access to a number of devices by default (disk,network,etc)
  - Only the tasks that have granted allocation on the **GRES** devices will be allowed to have access on them.
  - Tasks with no granted allocation upon **GRES** devices will not be able to use them.

task/cgroup plugin : **devices** subsystem

Configuration

1) Gres

- Configure gres plugin in the **slurm.conf**
- Create a **gres.conf** file on each computing node describing its resources

2) Cgroup Devices

- Configure **cgroup.conf** file to constrain devices
- Create file **allowed_devices.conf** file on each computing node listing the devices that the system should allow by default for all tasks

task/cgroup plugin : **devices** subsystem

GRES Configuration Example

```
[root@mordor cgroup]# egrep "Gres" /etc/slurm/slurm.conf
GresTypes=gpu
NodeName=mordor NodeAddr=127.0.0.1 Gres=gpu:2 Sockets=1 Cor...
```

```
[root@mordor cgroup]# cat /etc/slurm/slurm.conf
Name=gpu File=/dev/nvidia1
Name=gpu File=/dev/nvidia2
```

Note:
To declare different slurm.conf between nodes and controller you need to use option Debug_Flags=NO_CONF_HASH

task/cgroup plugin : **devices** subsystem

Cgroup Devices Configuration Example

```
[root@mordor cgroup]# egrep "Devices" /usr/local/etc/cgroup.conf
ConstrainDevices=yes
AllowedDevicesFile="/usr/local/etc/allowed_devices.conf"
```

```
[root@mordor cgroup]# cat /usr/local/etc/allowed_devices.conf
/dev/sda*
/dev/null
/dev/zero
/dev/urandom
/dev/cpu/*/*
```

task/cgroup plugin : **devices** subsystem

Cgroup Devices Logic as implemented in task plugin

1) Initialization phase (information collection gres.conf file, major, minor, etc)

2) Allow all devices that should be allowed by default (allowed_devices.conf)

3) Lookup which gres devices are allocated for the job

- Echo allowed gres devices to devices.allow file
- Echo denied gres devices to devices.deny file

4) Execute **2** and **3** for job and steps tasks (different hierearchy level in cgroups)

task/cgroup plugin : **devices** subsystem

```
[mat@slacklap slurm]$ srun -n1 –gres=gpu:1 sleep 100
```

```
[root@slacklap ~]# tail -f /var/log/slurmd.mordor.log |grep device
[2011-09-18T00:41:33] [327.0] Default access allowed to device c 202:0 rwm
[2011-09-18T00:41:33] [327.0] parameter 'devices.allow' set to 'c 202:0 rwm' for
    '/cgroup/devices/uid_0/job_327step_0'
[2011-09-18T00:41:33] [327.0] Default access allowed to device c 1:5 rwm
[2011-09-18T00:41:33] [327.0] parameter 'devices.allow' set to 'c 1:5 rwm' for
    '/cgroup/devices/uid_0/job_327/step_0'
```

```
[gohn@mordor ~]$ cat /cgroup/devices/uid_500/job_335/step_0/devices.list
c 202:0 rwm
c 1:5 rwm
b 8:1 rwm
[gohn@mordor ~]$ cat /cgroup/devices/uid_500/job_335/step_0/tasks
2980
2984
```

task/cgroup plugin : **devices** subsystem

Future Improvements

- For the moment only jobs and steps hierarchies have their devices constrained. Future goal is to constrain devices on the user level hierarchy with the help of the PAM plugin

- Improvements in cgroup/devices subsystem have been proposed to the kernel developers. The most important is related with the function of devices as whitelist and not as blacklist. The second would ease the procedure and no allowed_devices.conf file would be needed.

jobacct_gather/cgroup plugin

- New version of jobacct_gather plugin that uses cgroups to collect CPU and memory usage statistics for jobs and steps
- Potential for major performance improvement compared to jobacct_gather/linux plugin due to automatic addition of descendent processes to task groups
- Uses cpuacct and memory subsystems
- Current status is experimental due to limitations in data provided by cgroups
- Bug reported: cpuacct.stat state object reports CPU time, not CPU cycles as documented

Ongoing Works: SLURM cgroups and PAM integration

A **PAM module** to leverage the user cgroup and help system daemons to bind user 's tasks to the locally allocated resources only

- OpenSSH will use that PAM module to only allow remote log in to allocated resources
- MPI implementations not aware of SLURM (using ssh, like IntelMPI) could be confined

Possible Improvements: **devices** subsystem

- Improvements in cgroup/devices subsystem have been proposed to the kernel developers. One of them is related with the function of devices as whitelist and not as both white and black-list. This would ease the procedure and no allowed_devices.conf file would be required.

Future Research Works

Limit the usage of disk and network bandwidth

- Control access to **I/O on hard disks** for tasks in cgroups through **blkio** subsystem
  - By specifying relative proportion (blkio.weight) of I/O access of devices available to a cgroup through the blkio.weight parameter with range from 100 to 1000
- Limit the **network bandwidth** for tasks in cgroups through **net_cls** subsystem
  - By specifying particular ids (net_cls.classids) and configure them appropriately through the filtering capabilities of the Linux network stack (tc command) to provide particular network bandwith to each cgroup
- Implementation as new parameters in the **task cgroup plugin**
- **Issues: net_cls** currently works only for ethernet (not for infiniband) and **blkio** would work only for local hard disks (not for Lustre)

Future Research Works

Monitor and report the usage of additional resources

- Monitor and report **I/O access on hard disks** for tasks in cgroups **blkio subsystem**
  - Report may contain I/O time and I/O bytes transferred
  - How to monitor on NFS or Lustre systems?
- How to monitor and report network **usage** ?
- How to monitor and report energy consumption?
  - Resource Individual Power consumption
  - Energy consumption per process and per resource

References

Cgroups integration upon SLURM, involved developers:

-Matthieu Hautreux (CEA, France)

-Martin Perry (BULL, USA)

-Yiannis Georgiou (BULL, France)

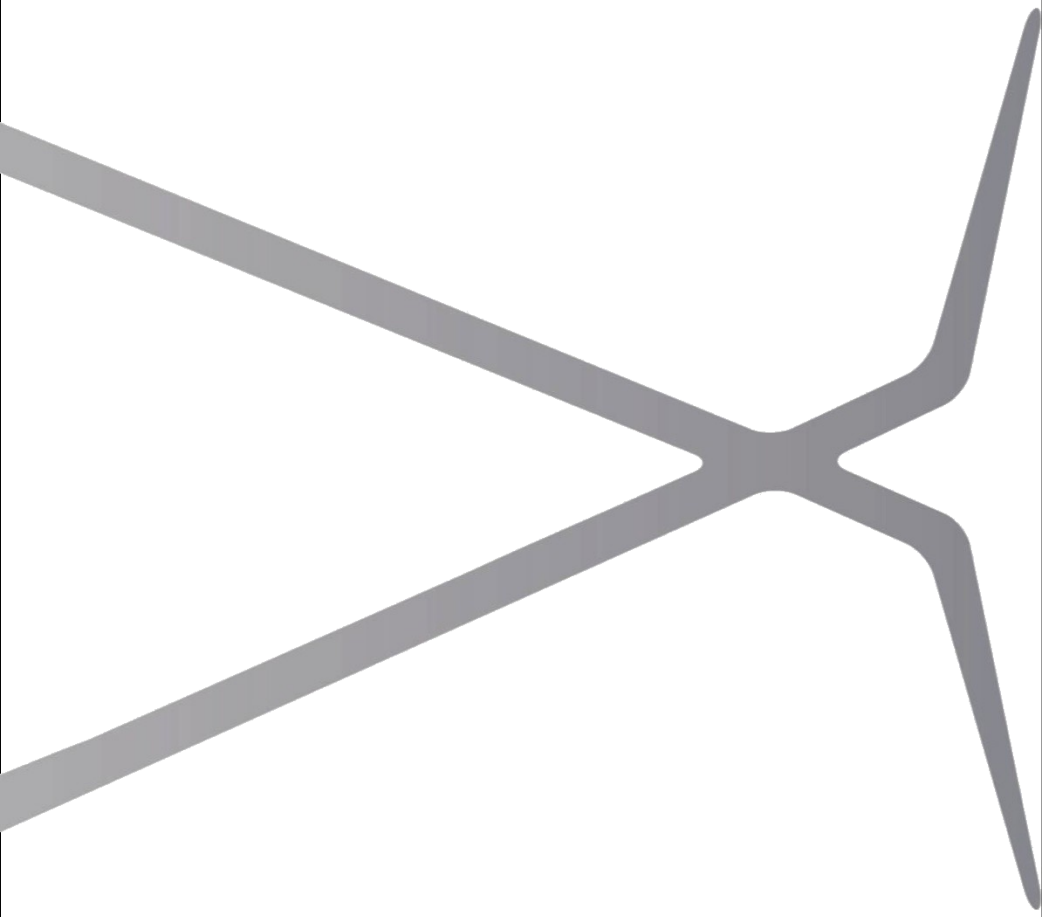
-Mark Grondona (LLNL, USA)

-Morris Jette (SchedMD, USA)

-Danny Auble (SchedMD, USA)

SLURM source code:

git clone git://github.com/SchedMD/slurm.git



bullx

instruments for innovation