Resource Management with Linux Control Groups in HPC Clusters

Yiannis Georgiou
email: yiannis.georgiou@bull.fr

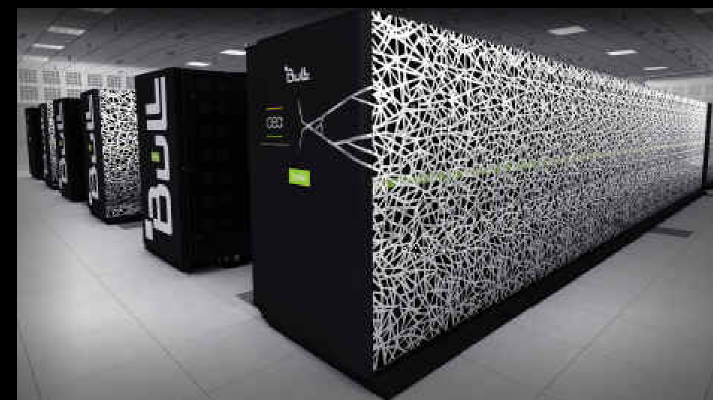
Outline

- Introduction
  - Motivations
  - Linux cgroups
  - SLURM Resource and Job Management System
- Cgroups integration upon SLURM
  - Basic API for cgroups usage
- Cgroups subsystems support for SLURM
  - Organization and Configuration
  - Usage Examples
  - Linux Community Interactions
- Current Ongoing work
- Future Improvements and Perspectives

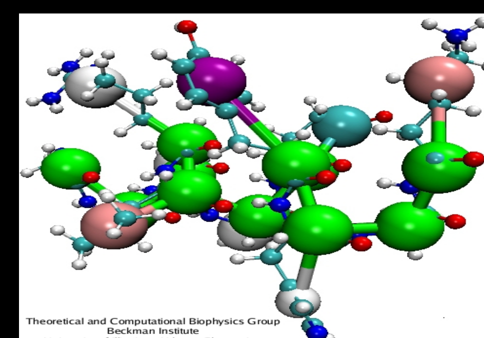
High Performance Computing

High Performance Computing is defined by:

Infrastructures:
Supercomputers, Clusters, Grids, Peer-to-Peer Systems and lately Clouds



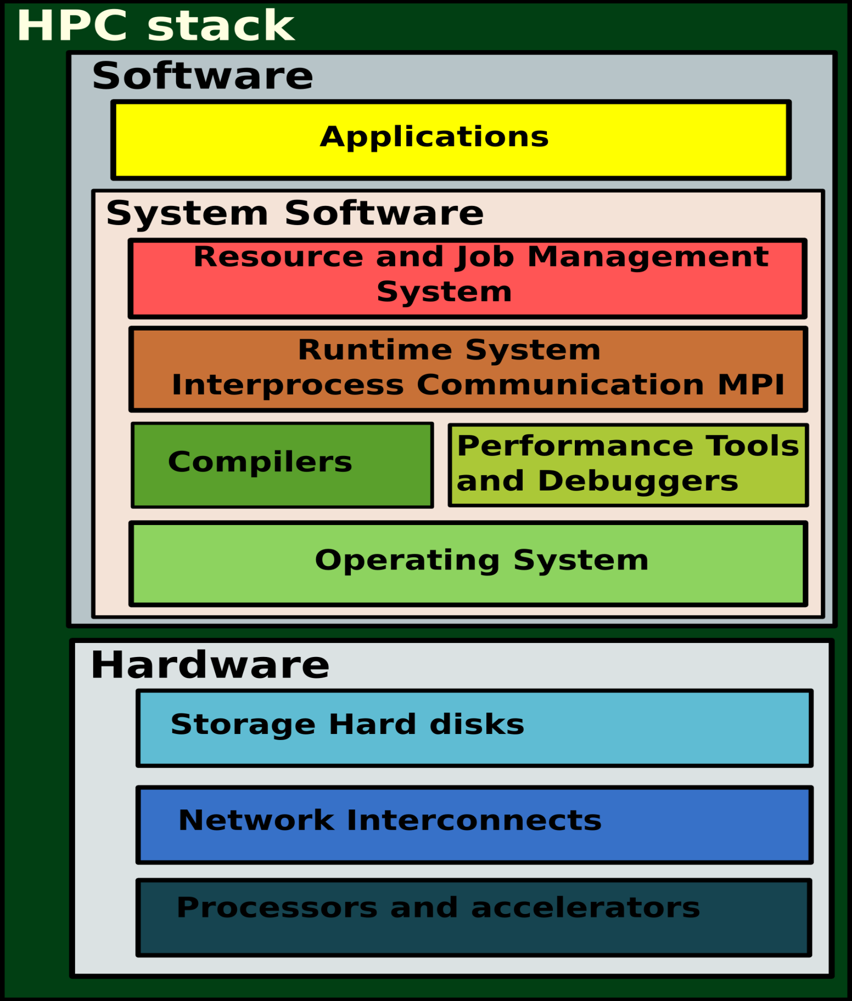
Applications:
Climate Prediction,Protein Folding, Crash simulation, High-Energy Physics, Astrophysics, Animation for movie and video game productions



High Performance Computing

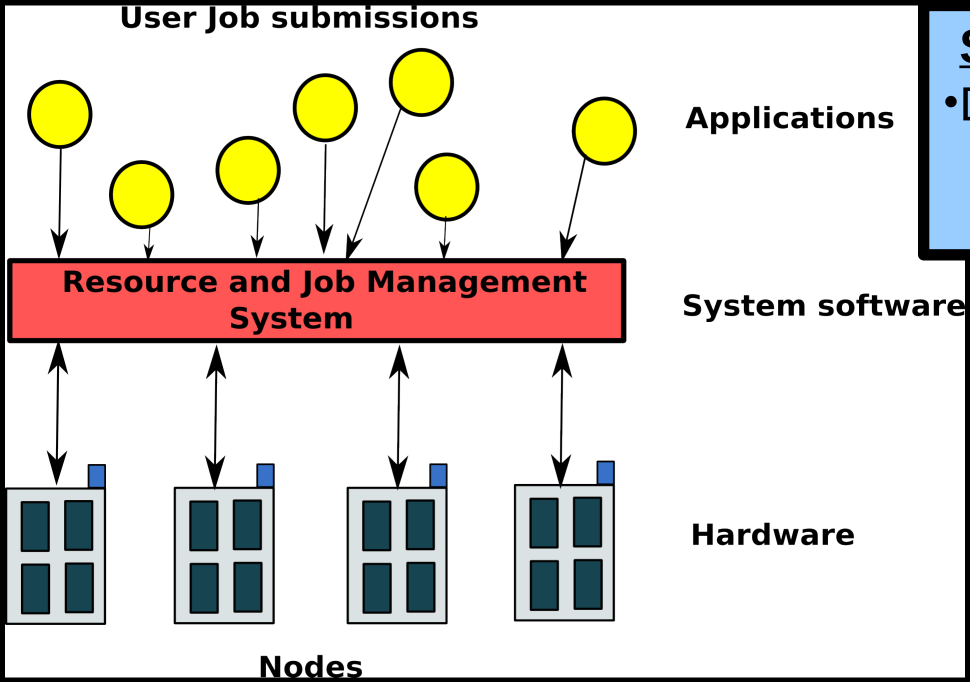
System Software:

Operating System, Runtime System, Resource Management, I/O System, Interfacing to External Environments



Resource and Job Management Systems

The goal of a Resource and Job Management System (RJMS) is to satisfy users' demands for computation and assign resources to user jobs with an efficient manner.

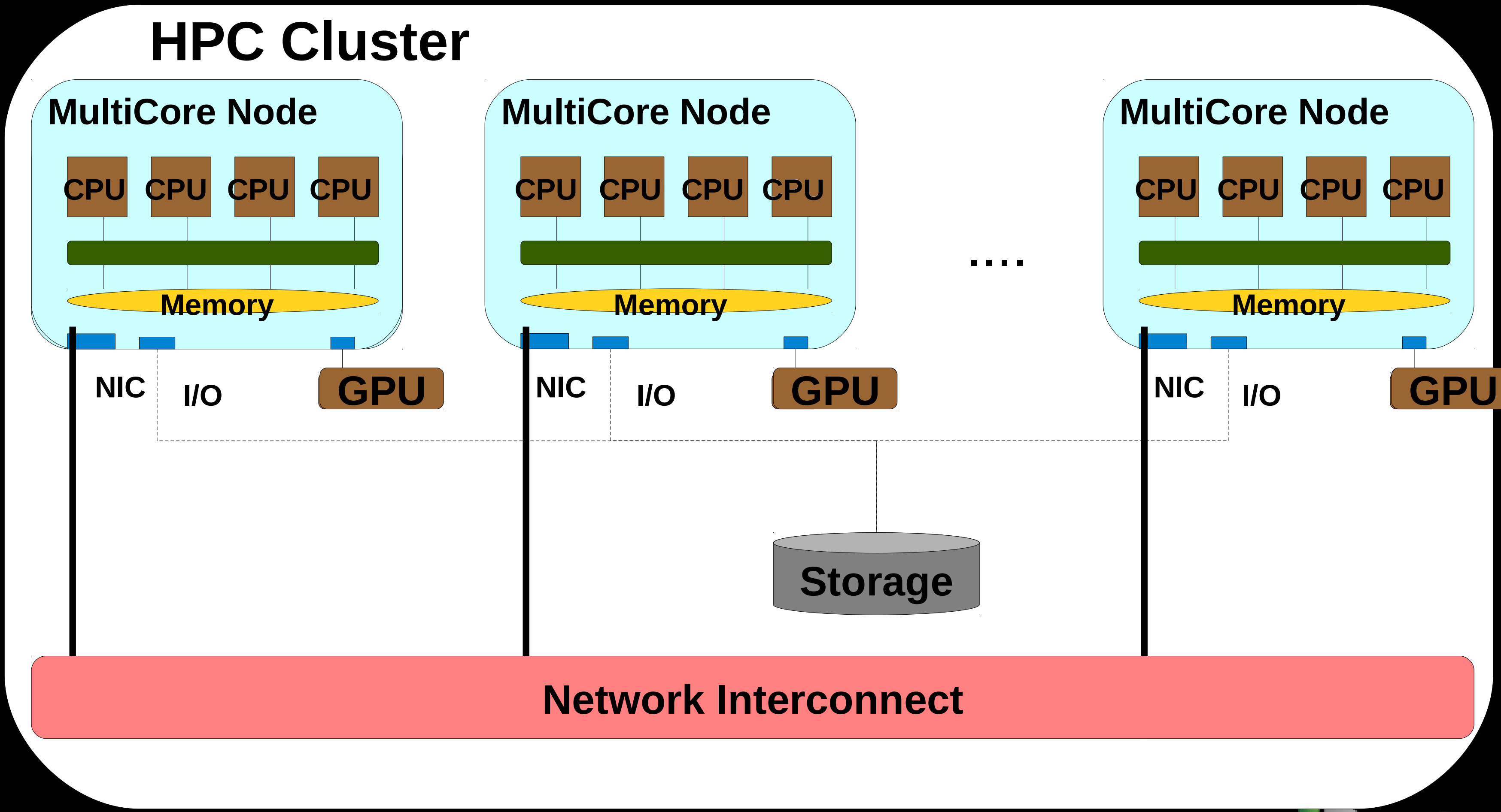


Strategic position in HPC stack

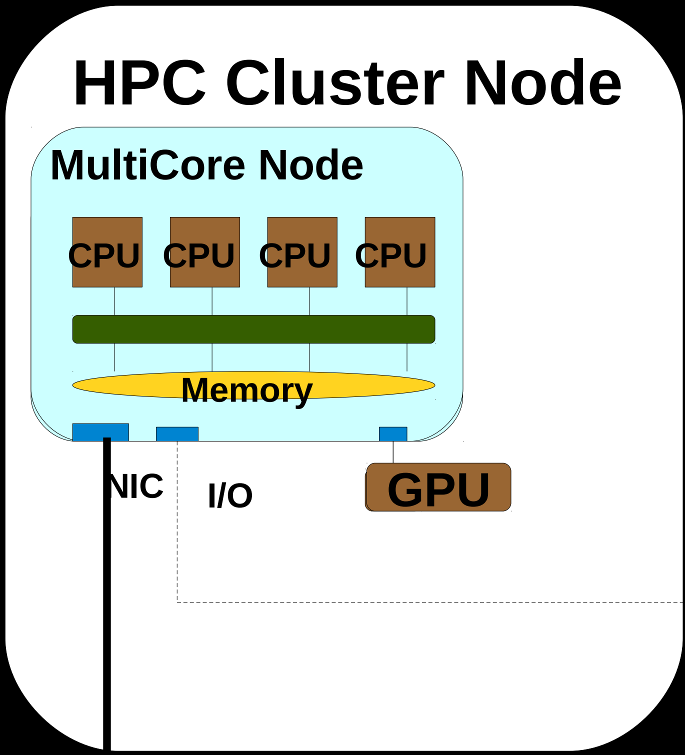
- Direct and constant knowledge:
  - of **Resources** characteristics/states
  - of **Jobs** needs/states

Resource Management in HPC

-HPC Clusters nowadays: Proliferation of Resources



Resource Management in HPC



Cluster Node Evolutions:

- Increase of number of CPUs/Cores per node
- Deeper memory hierarchies (SMP, NUMA, etc)
- Multiple Network Interface Cards and GPUs
- Bandwidth of Network and I/O seen as extra resources

How can the RJMS of HPC clusters provide efficient and robust Resource Management ?

In terms of: 1) Allocation 2) Limiting 3) Monitoring

Issues with no Task Confinement upon the allocated resources on HPC clusters

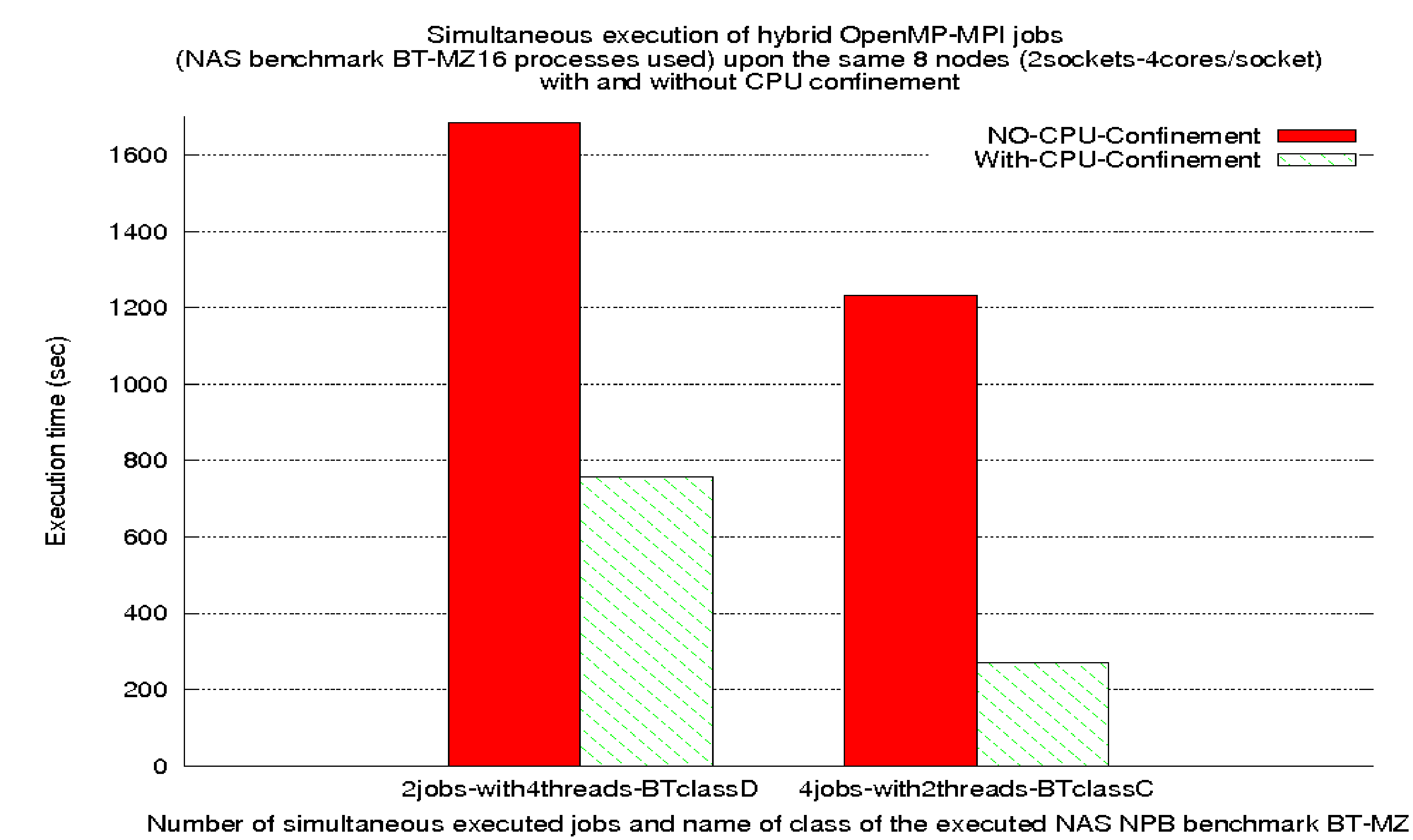
- SMP system without some means of CPU placement, any task can run on any CPU.

  This may cause CPU idleness while other CPUs are shared and system time spent on migrating tasks between processors
- NUMA system, any memory page can be allocated on any node.

  This can cause both poor cache locality and poor memory access times.

Issues with no Task Confinement upon the allocated resources on HPC clusters

Execution of NAS benchmarks (MPI-OpenMP) upon 8 nodes SMP cluster (2sockets-4cores/socket) with and without CPU confinement



Issues with no Task Confinement upon the allocated resources on HPC clusters

CPU instant utilization during execution of 2 BT-MZ jobs with 16 tasks and 4 threads per task without CPU confinement

```
top - 13:32:36 up  1:21,  1 user,  load average: 7.05, 5.18, 6.79
Tasks: 256 total,   9 running, 247 sleeping,   0 stopped,   0 zombie
Cpu0  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  18395656k total,  3346396k used, 15049260k free,    15764k buffers
Swap:  1022752k total,        0k used,  1022752k free,   114104k cached
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  P COMMAND
 3582 georgioy  20   0  920m 713m 3956 R 54.8  4.0   0:45.34 3 bt-mz.D.16
 3581 georgioy  20   0  921m 717m 4192 R 52.5  4.0   0:45.47 1 bt-mz.D.16
 3592 georgioy  20   0  921m 713m 3924 R 51.2  4.0   0:43.02 2 bt-mz.D.16
 3577 georgioy  20   0  920m 717m 3940 R 50.8  4.0   0:44.81 0 bt-mz.D.16
 3578 georgioy  20   0  921m 713m 3924 R 50.2  4.0   0:45.37 3 bt-mz.D.16
 3594 georgioy  20   0  920m 717m 3940 R 48.5  4.0   0:43.48 0 bt-mz.D.16
 3598 georgioy  20   0  921m 717m 4192 R 48.2  4.0   0:43.14 1 bt-mz.D.16
 3597 georgioy  20   0  920m 713m 3956 R 43.9  4.0   0:43.18 2 bt-mz.D.16
    1 root      20   0 21336 1548 1280 S  0.0  0.0   0:03.60 5 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 5 kthreadd
```

CPUs are shared between jobs

While there are idle CPUs

Advantages: cgroups support for HPC

- To guarantee that every consumed resources is consumed the way it's planned to be
  - leveraging Linux latest features in terms of process control and resource management
  - Enabling node sharing
- While enhancing the connection with Linux systems
  - Improve **tasks isolation** upon resources
  - Improve **efficiency** of resource management activities (e.g., process tracking, collection of accounting statistics)
  - Improve **robustness** (e.g. more reliable cleanup of jobs)
- And simplifying the addition of **new controlled resources and features**
  - prospective management of network and I/O as individual resources

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. montior resource usage for processes)
- ***Control*** (i.e. suspending and resuming processes)

Cgroups Model and Concepts

Model

Cgroups **similar** to Linux processes:

- Hierarchical
- Inheritance of attributes from parent to child

but **different** because:

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

Concepts

Cgroup – a group of processes with the same characteristics

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

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)

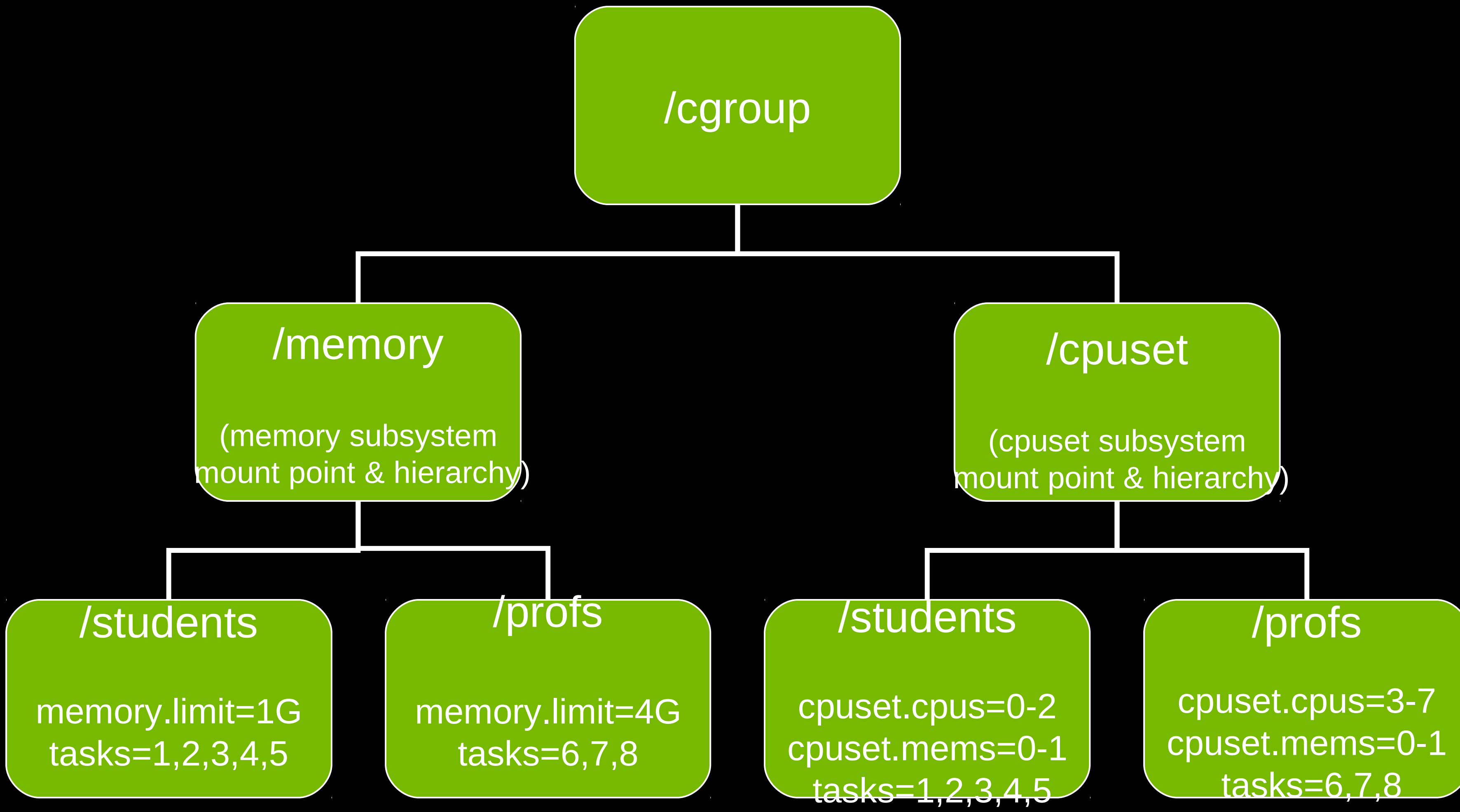
freezer subsystem

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 acces

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:~]# echo 0-2 > /cgroup/cpuset/students/cpuset.cpus
[root@mordor:~]# echo 0 > /cgroup/cpuset/students/cpuset.mems
[root@mordor:~]# echo $PIDS_st > /cgroup/cpuset/students/tasks
[root@mordor:~]# echo 3-7 > /cgroup/cpuset/profs/cpuset.cpus
[root@mordor:~]# echo 1 > /cgroup/cpuset/profs/cpuset.mems
[root@mordor:~]# echo $PIDS_pr > /cgroup/cpuset/profs/tasks
```

SLURM Resource and Job Management System for Linux Clusters

SLURM **open-source** Resource and Job Management System

- Developed since 2003, initially in LLNL and then SchedMD since 2011
- Multiple enterprises and research centers are contributing to the project (LANL,CEA,HP,BULL,BSC, etc)
- Large international community
  - Contributions (various external software and standards are integrated upon SLURM)
- Used on a lot of worlds largest supercomputers, amongst which:
  - Tianhe-1A with 2.5 Petaflop 2nd of Top500 in 2011
  - Tera100 with 1.25 Petaflop 1st European of Top500 in 2011
  - ...planned IBM BlueGene/Q with 20 Petaflop, for 2012

SLURM: A Flexible and Scalable RJMS

- **Portable:** written in C with a GNU autoconf configuration engine.
- **Modular:** Based on a plugin mechanism used to support different kind of scheduling policies, interconnects, libraries, etc
- **Robust:** highly tolerant of system failures, including failure of the node executing its control functions.
- **Scalable:** designed for up to 65,536 nodes and hundreds of thousands of processors and can sustain a throughput rate of over 120,000 jobs per hour with bursts of job submissions at several times that rate.

Cgroups implementation upon SLURM

- A common API to manage cgroup hierarchies, directories and files
  - src/common/xcgroup.{h,c}
  - src/common/xcgroup_read_config.{h,c}
- A uniform syntax to declare slurm related cgroup subsystems directories
  - %cgroup_subsys_mount_point%/uid_%uid/job_%jobid/step_%stepid/
- A dedicated cgroup release_agent and subsystems release_agent naming schema
  - Lock/Unlock cgroup hierarchy when managing slurm related cgroups to avoid race conditions
  - Update uid_%uid entry to match subdirectory configurations
- 2 plugins that add cgroup related features to slurmd
  - Proctrack/cgroup : to track/suspend/resume job's tasks
  - Task/cgroup : to confine tasks to the allocated resources

SLURM Cgroups API

Ease cgroup init, directories and files management

- slurm_cgroup_conf_t
  - Stores cgroup related conf
- xcgroup_ns_t
  - Structure associated to 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

Process Tracking with Cgroups

Track job processes using the freezer subsystem

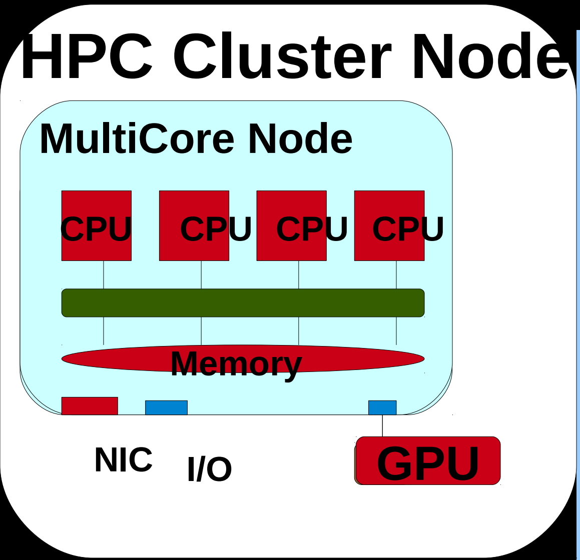
- Every spawned process is tracked
  - Automatic inheritance of parent's cgroup
  - No way to escape the container
- Every processes can be frozen
  - Using the Thawed|Frozen state of the subsystem
  - No way to avoid the freeze action

Cgroup **Proctrack** plugin: **freezer** subsystem

[mat@leaf slurm]$ srun sleep 300

```
[root@leaf ~]# cat /cgroup/freezer/uid_500/job_53/step_0/freezer.state
THAWED
[root@leaf ~]# scontrol suspend 53
[root@leaf ~]# ps -ef f | tail -n 2
root     15144     1  0 17:10 ?        Sl     0:00 slurmstepd: [53.0]
mat      15147 15144  0 17:10 ?        T      0:00  \_ /bin/sleep 300
[root@leaf ~]# cat /cgroup/freezer/uid_500/job_53/step_0/freezer.state
FREEZING
[root@leaf ~]# scontrol resume 53
[root@leaf ~]# ps -ef f | tail -n 2
root     15144     1  0 17:10 ?        Sl     0:00 slurmstepd: [53.0]
mat      15147 15144  0 17:10 ?        S      0:00  \_ /bin/sleep 300
[root@leaf ~]# cat /cgroup/freezer/uid_500/job_53/step_0/freezer.state
THAWED
[root@leaf ~]#
```

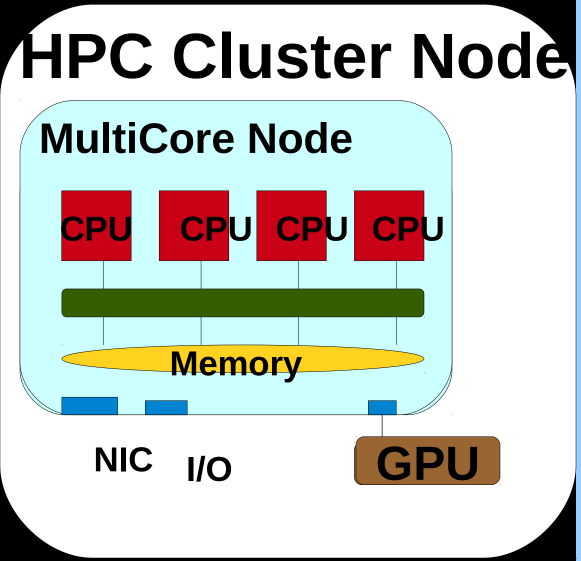
Task confinement for allocated resources



Constrain jobs tasks to the allocated resources

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

Task confinement for cpus



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)
  - Automatic block and cyclic distribution of tasks

Cgroup **Task** plugin : **cpuset** subsystem

```
[mat@leaf slurm]$ salloc --exclusive srun -n1 --cpu_bind=none sleep 3000
salloc: Granted job allocation 55
```

```
[root@leaf ~]# egrep "Cores|Affinity" /etc/slurm/cgroup.conf
ConstrainCores=yes
TaskAffinity=yes
[root@leaf ~]# tail -f /var/log/slurmd.leaf10.log |grep task/cgroup
[2011-09-16T17:24:59] [55.0] task/cgroup: now constraining jobs allocated cores
[2011-09-16T17:24:59] [55.0] task/cgroup: loaded
[2011-09-16T17:24:59] [55.0] task/cgroup: job abstract cores are '0-31'
[2011-09-16T17:24:59] [55.0] task/cgroup: step abstract cores are '0-31'
[2011-09-16T17:24:59] [55.0] task/cgroup: job physical cores are '0-31'
[2011-09-16T17:24:59] [55.0] task/cgroup: step physical cores are '0-31'
[2011-09-16T17:24:59] [55.0] task/cgroup: task[0] is requesting no affinity
```

Cgroup **Task** plugin : **cpuset** subsystem

```
[mat@leaf slurm]$ salloc --exclusive srun -n1 –exclusive
  --cpu_bind=none sleep 3000
salloc: Granted job allocation 56
```

```
 [root@leaf ~]# egrep "Cores|Affinity" /etc/slurm/cgroup.conf
ConstrainCores=yes
TaskAffinity=yes
[root@leaf ~]# tail -f /var/log/slurmd.leaf10.log |grep task/cgroup
[2011-09-16T17:29:25] [56.0] task/cgroup: now constraining jobs allocated cores
[2011-09-16T17:29:25] [56.0] task/cgroup: loaded
[2011-09-16T17:29:25] [56.0] task/cgroup: job abstract cores are '0-31'
[2011-09-16T17:29:25] [56.0] task/cgroup: step abstract cores are '0'
[2011-09-16T17:29:25] [56.0] task/cgroup: job physical cores are '0-31'
[2011-09-16T17:29:25] [56.0] task/cgroup: step physical cores are '0'
[2011-09-16T17:29:25] [56.0] task/cgroup: task[0] is requesting no affinity
```

Cgroup **Task** plugin : **cpuset** subsystem

```
[mat@leaf slurm]$ salloc --exclusive srun -n1 --cpu_bind=cores sleep 3000
salloc: Granted job allocation 57
```

```
[root@leaf ~]# egrep "Cores|Affinity" /etc/slurm/cgroup.conf
ConstrainCores=yes
TaskAffinity=yes
[root@leaf ~]# tail -f /var/log/slurmd.leaf10.log |grep task/cgroup
[2011-09-16T17:31:17] [57.0] task/cgroup: now constraining jobs allocated cores
[2011-09-16T17:31:17] [57.0] task/cgroup: loaded
[2011-09-16T17:31:17] [57.0] task/cgroup: job abstract cores are '0-31'
[2011-09-16T17:31:17] [57.0] task/cgroup: step abstract cores are '0-31'
[2011-09-16T17:31:17] [57.0] task/cgroup: job physical cores are '0-31'
[2011-09-16T17:31:17] [57.0] task/cgroup: step physical cores are '0-31'
[2011-09-16T17:31:17] [57.0] task/cgroup: task[0] is requesting core level binding
[2011-09-16T17:31:17] [57.0] task/cgroup: task[0] using Core granularity
[2011-09-16T17:31:17] [57.0] task/cgroup: task[0] taskset '0x00000001' is set
```

Cgroup **Task** plugin : **cpuset** subsystem

```
[mat@leaf slurm]$ salloc --exclusive srun -n1 --cpu_bind=socket sleep 3000
salloc: Granted job allocation 58
```

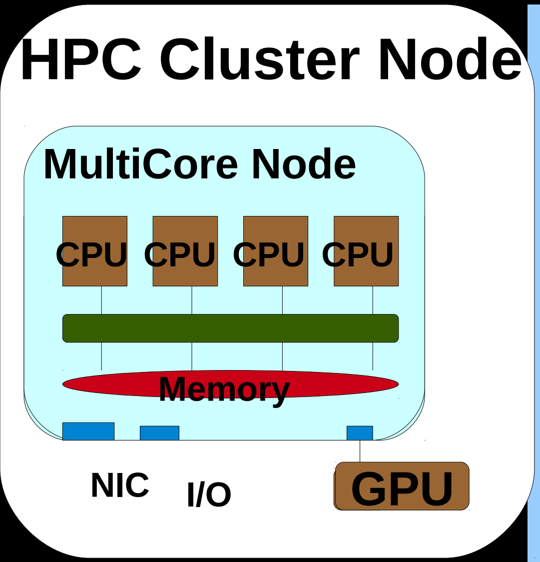
```
[root@leaf ~]# egrep "Cores|Affinity" /etc/slurm/cgroup.conf
ConstrainCores=yes
TaskAffinity=yes
[root@leaf ~]# tail -f /var/log/slurmd.leaf10.log |grep task/cgroup
[2011-09-16T17:33:31] [58.0] task/cgroup: now constraining jobs allocated cores
[2011-09-16T17:33:31] [58.0] task/cgroup: loaded
[2011-09-16T17:33:31] [58.0] task/cgroup: job abstract cores are '0-31'
[2011-09-16T17:33:31] [58.0] task/cgroup: step abstract cores are '0-31'
[2011-09-16T17:33:31] [58.0] task/cgroup: job physical cores are '0-31'
[2011-09-16T17:33:31] [58.0] task/cgroup: step physical cores are '0-31'
[2011-09-16T17:33:31] [58.0] task/cgroup: task[0] is requesting socket level binding
[2011-09-16T17:33:31] [58.0] task/cgroup: task[0] using Socket granularity
[2011-09-16T17:33:31] [58.0] task/cgroup: task[0] taskset '0x00000003' is set
```

Cgroup **Task** plugin : **cpuset** subsystem

[mat@leaf slurm]$ salloc --exclusive srun -n2 --cpu_bind=socket sleep 3000
salloc: Granted job allocation 60

```
[root@leaf ~]# egrep "Cores|Affinity" /etc/slurm/cgroup.conf
ConstrainCores=yes
TaskAffinity=yes
[root@leaf ~]# tail -f /var/log/slurmd.leaf10.log |grep task/cgroup[2011-09-16T17:36:18] [60.0] task/cgroup:
 now constraining jobs allocated cores
[2011-09-16T17:36:18] [60.0] task/cgroup: loaded
[2011-09-16T17:36:18] [60.0] task/cgroup: job abstract cores are '0-31'
[2011-09-16T17:36:18] [60.0] task/cgroup: step abstract cores are '0-31'
[2011-09-16T17:36:18] [60.0] task/cgroup: job physical cores are '0-31'
[2011-09-16T17:36:18] [60.0] task/cgroup: step physical cores are '0-31'
[2011-09-16T17:36:18] [60.0] task/cgroup: task[0] is requesting socket level binding
[2011-09-16T17:36:18] [60.0] task/cgroup: task[1] is requesting socket level binding
[2011-09-16T17:36:18] [60.0] task/cgroup: task[1] using Core granularity
[2011-09-16T17:36:18] [60.0] task/cgroup: task[1] higher level Socket found
[2011-09-16T17:36:18] [60.0] task/cgroup: task[1] taskset '0x00000003' is set
[2011-09-16T17:36:18] [60.0] task/cgroup: task[0] using Core granularity
[2011-09-16T17:36:18] [60.0] task/cgroup: task[0] higher level Socket found
[2011-09-16T17:36:18] [60.0] task/cgroup: task[0] taskset '0x00000003' is set
```

Task confinement for memory : **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”
  - Else, let steps use job's allocated amount
- Both RSS and swap are monitored
- Trigger OOM killer on the cgroup's tasks when reaching limits
- Tolerant mechanism
  - AllowedRAMSpace , AllowedSwapSpace percents

Cgroup **Task** plugin : **memory** subsystem

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

```
[root@leaf ~]# 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@leaf slurm]$ salloc --exclusive --mem-per-cpu 100 srun –exclusive -n1 sleep
 3000
salloc: Granted job allocation 68
```

```
[root@leaf ~]# 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
```

Cgroup **Task** plugin : **memory** subsystem OOM killer usage

```
[mat@leaf slurm]$ salloc --exclusive --mem-per-cpu 40 srun -n2 ./malloc
salloc: Granted job allocation 268
```

```
slurmd[berlin27]: Step 268.0 exceeded 1310720 KB memory limit,
  being killed
srun: Exceeded job memory limit
srun: Job step aborted: Waiting up to 2 seconds for job step to finish.
slurmd[berlin27]: *** STEP 268.0 KILLED AT 2012-03-31T15:50:36
  WITH SIGNAL 9 ***
srun: error: berlin27: tasks 0,1: Killed
```

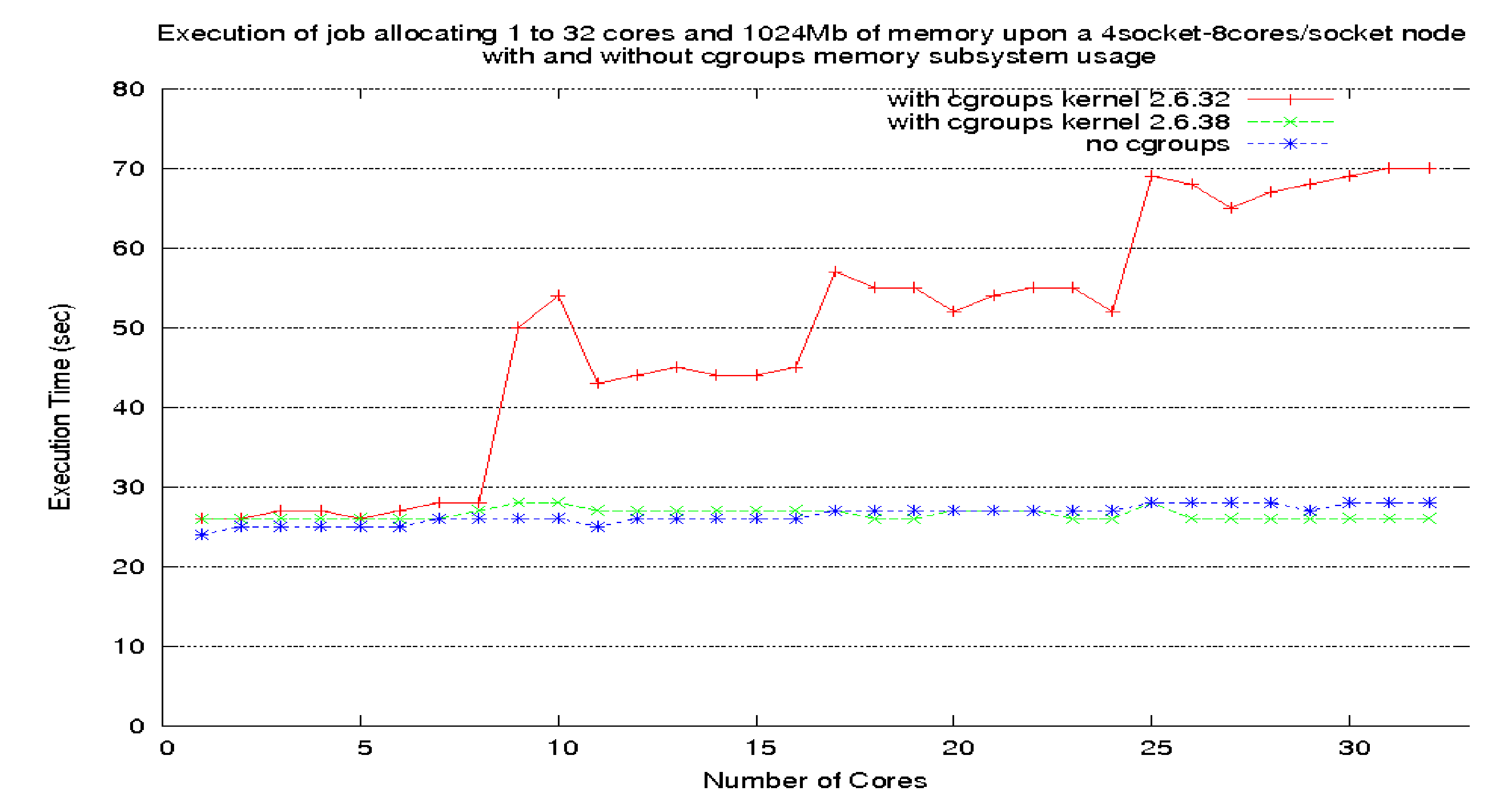
Cgroup **Task** plugin : **memory** subsystem Problems Limitations

Performances penalities on some systems

- Depending on the kernel/cgroup version
- Depending on the NUMA architecture of the nodes

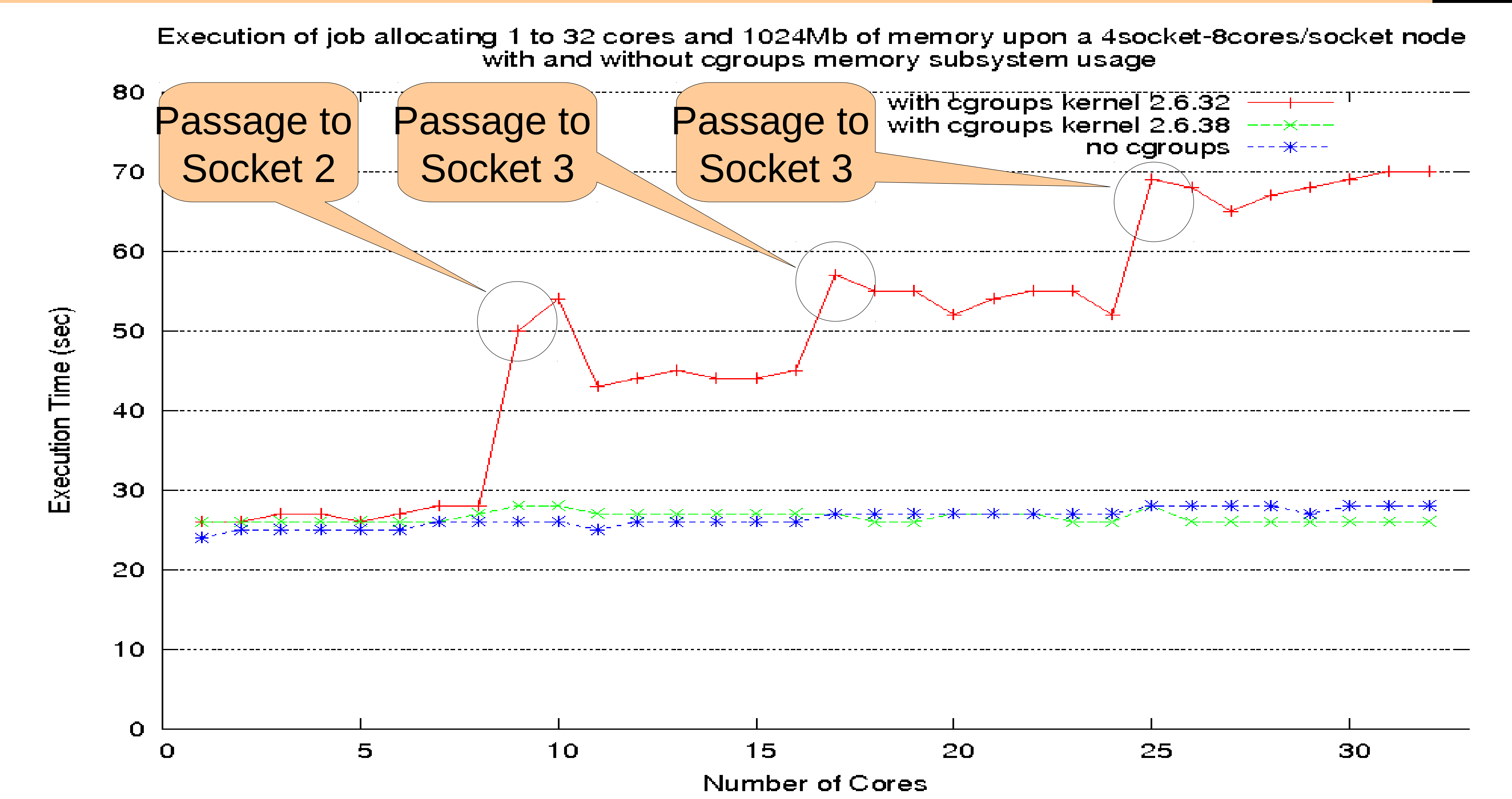
Cgroup **Task** plugin : **memory** subsystem
Problems Limitations

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



Cgroup **Task** plugin : **memory** subsystem Problems Limitations

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



Cgroup **Task** 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]

Cores racing for spinlock

Cgroup **Task** plugin : **memory** subsystem Improvements

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

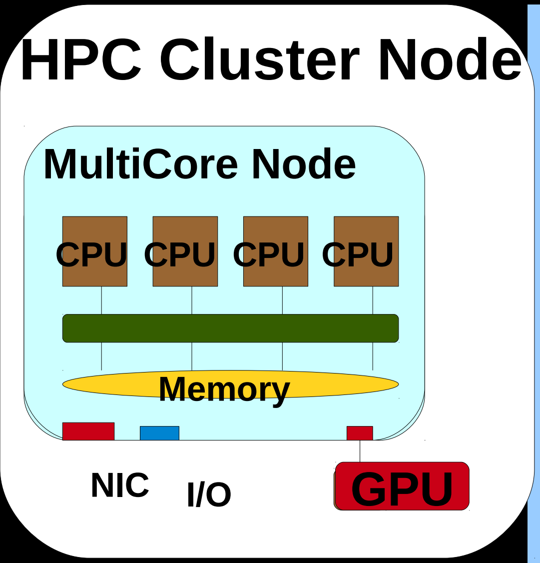
Problem corrected by cgroups maintainers

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

Tasks confinement for devices: **devices** subsystem



Constrain jobs tasks to the allocated system devices

- Based on the **GRES** plugin for generic resources allocation (NIC, GPUs, etc) and built upon the cgroup task plugin
  - Each task is allowed to access to a number of devices by default
  - 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.

Cgroup **Task** plugin : **devices** subsystem

Cgroup Devices Configuration Example

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

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

Cgroup **Task** 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

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

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

Cgroups **devices** subsystem : Usage Example

```
[root@mordor cgroup]# egrep "Gres" /etc/slurm/slurm.conf
GresTypes=gpu
NodeName=cuzco[57,61] Gres=gpu:2 Procs=8 Sockets=2 CoresPerSocket=4
```

```
[root@cuzco51]# cat /etc/slurm/allowed_devices.conf
/dev/sda*
/dev/null
```

```
[gohn@cuzco0]$ cat gpu_test.sh
#!/bin/sh
sleep 10
echo 0 > /dev/nvidia0
echo 0 > /dev/nvidia1
```

Cgroups **devices** subsystem : Usage Example

[gohn@cuzco0]$ srun -n1 –gres=gpu:1 -o output ./gpu_test.sh

```
[root@cuzco51 ~]# tail -f /var/log/slurmd.cuzco51.log
[2011-09-20T03:10:02] [22.0] task/cgroup: manage devices jor job '22'
[2011-09-20T03:10:02] [22.0] device : /dev/nvidia0 major 195, minor 0
[2011-09-20T03:10:02] [22.0] device : /dev/nvidia1 major 195, minor 1
[2011-09-20T03:10:02] [22.0] device : /dev/sda2 major 8, minor 2
[2011-09-20T03:10:02] [22.0] device : /dev/sda1 major 8, minor 1
[2011-09-20T03:10:02] [22.0] device : /dev/sda major 8, minor 0
[2011-09-20T03:10:02] [22.0] device : /dev/null major 1, minor 3
[2011-09-20T03:10:02] [22.0] Default access allowed to device b 8:2 rwm
[2011-09-20T03:10:02] [22.0] parameter 'devices.allow' set to 'b 8:2 rwm' for '/cgroup/devices/uid_50071/job_22/step_0'
[2011-09-20T03:10:02] [22.0] Default access allowed to device b 8:1 rwm
[2011-09-20T03:10:02] [22.0] parameter 'devices.allow' set to 'b 8:1 rwm' for '/cgroup/devices/uid_50071/job_22/step_0'
[2011-09-20T03:10:02] [22.0] Default access allowed to device b 8:0 rwm
[2011-09-20T03:10:02] [22.0] parameter 'devices.allow' set to 'b 8:0 rwm' for '/cgroup/devices/uid_50071/job_22/step_0'
[2011-09-20T03:10:02] [22.0] Default access allowed to device c 1:3 rwm
[2011-09-20T03:10:02] [22.0] parameter 'devices.allow' set to 'c 1:3 rwm' for '/cgroup/devices/uid_50071/job_22/step_0'
[2011-09-20T03:10:02] [22.0] Allowing access to device c 195:0 rwm
[2011-09-20T03:10:02] [22.0] parameter 'devices.allow' set to 'c 195:0 rwm' for '/cgroup/devices/uid_50071/job_22/step_0'
[2011-09-20T03:10:02] [22.0] Not allowing access to device c 195:1 rwm
[2011-09-20T03:10:02] [22.0] parameter 'devices.deny' set to 'c 195:1 rwm' for '/cgroup/devices/uid_50071/job_22/step_0'
```

Cgroups **devices** subsystem : Usage Example

```
[root@cuzco51 ~]# cat /cgroup/devices/uid_50071/job_22/step_0/tasks
4875
4879
4882
[root@cuzco51 ~]# cat /cgroup/devices/uid_50071/job_22/step_0/devices.list
b 8:2 rwm
b 8:1 rwm
b 8:0 rwm
c 1:3 rwm
c 195:0 rwm
```

```
[gohn@cuzco0]$ cat output
/home/GPU/./gputest.sh: line 4: echo: write error: Invalid argument
/home/GPU/./gputest.sh: line 5: /dev/nvidia1: Operation not permitted
```

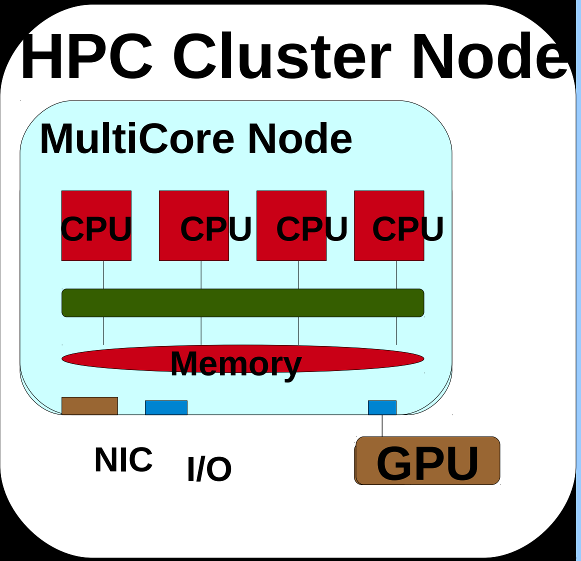
Cgroup **Task** plugin : **devices** subsystem Problems Limitations

Existing bug between NVIDIA API and cgroups devices

- The independent usage of a GPU through cgroups devices isolation is not allowed
- Open Bug RedHat Case Number: 00618885

```
//deny /dev/nvidia0 and allow /dev/nvidia1 and /dev/nvidiactl:
   echo c 195:0 rwm > /cgroup/devices/devices.deny
   echo c 195:1 rwm > /cgroup/devices/devices.allow
   echo c 195:255 rwm > /cgroup/devices/devices.allow
//try to get information of /dev/nvidia1
   nvidia-smi -g 1
  NVIDIA: could not open the device file /dev/nvidia0 (Operation not permitted).
   Failed to initialize NVML: Unknown Error
```

Monitoring Resource Usage:
cpuacct and **memory subsystems**



Monitoring cpu usage with cpuacct subsystem and memory usage with memory subsystem

- Implemented as a jobacct_gather plugin for SLURM
- Collects information concerning CPU time and Memory RSS consumed for each task of the cgroup
- Values reported as a new job characteristics in the accounting database of SLURM
- Values can be used for billing purposes
- Monitor per job energy consumption (not through cgroups)

Monitoring Resources: **cpuacct -memory** subsystems

```
[gohn@cuzco0]$ srun -n32  ./malloc
[gohn@cuzco0]$ sacct -j 167
```

```
   JobID    JobName Partition  MaxRSS   AveRSS   MaxPages  AvePages
 MinCPU       AveCPU  Elapsed  State  Ntasks  AllocCPUs  ExitCode
------------ ---------- -------------- -------------- ---------- ---------- ---------- ----------
   167.0    malloc  shared  61311K   57221K   239.24G  99893120K
 00:03.000    00:03.000  00:01:10  COMPLETED  32  32  0.0
```

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 monitor on NFS or Lustre systems?
- How Monitor and report network **usage** ?
- How 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 Grondnna (LLNL, USA)

-Morris Jette (SchedMD, USA)

-Danny Auble (SchedMD, USA)

SLURM source code:

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

THAKN YOU

Questions?