Resource Management with CGroups

Linux.conf.au 2011
Brisbane Australia

Steven Ellis
Red Hat Solution Architect
Ingram Micro New Zealand
Overview

- Control Group (cgroup)
  - Meet application SLAs by reducing resource contention and increasing predictability in performance. Resource controllers include:
    - CPU/CPUSSET
    - Memory
    - Network
    - I/O
Kernel Resource Management

- **Illustrative cgroup use cases**
  - Database workload dedicated 90%, background backup utility 10%
  - Virtualization hosting provider – allows QoS (quality of service guarantees based on pricepoint)

```
<table>
<thead>
<tr>
<th></th>
<th>Network</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>40% net</td>
<td></td>
<td>20%</td>
</tr>
<tr>
<td>40% net</td>
<td></td>
<td>20%</td>
</tr>
<tr>
<td>20% net</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Virt Guest A
- 50% CPU
- 50% Mem

Virt Guest B
- 25% CPU
- 25% Mem

Virt Guest C
- 25% CPU
- 25% Mem

Storage
- 60%
- 20%
Kernel resource management

• **Cgroup – Control group**
  • A control group provides a generic framework where several “resource controllers” can plug in and manage different resources of the system such as process scheduling, memory allocation, network traffic, or IO bandwidth.
    - Can be tracked to monitor system resource usage
    - Sysadmin can use tools to allow or deny these groups access to resources

• **Memory resource controller**
  • Isolates the memory behavior of a group of tasks – cgroup – from the rest of the system (including paging). It can be used to:
    - Isolate an application or a group of applications
    - Create a cgroup with limited amount of memory

• **Cgroup scheduler**
  • CFS – Hierarchical proportional fair scheduler (SCHED_OTHER)
  • Static priority scheduler with constant bandwidth limits (SCHED_FIFO)
Kernel resource management (continued)

• I/O controller
  • Designate portion of I/O bandwidth (based on controller queue depth)

• Network controller
  • Define classes & queues between generic network layer and NIC. Tagging packets with class identifier with different priorities, placing outbound packets in different queues for traffic shaping.

• libcgroup
  • Cgroup creation, deletion, move and configuration management.
  • Rules based automatic task placement, PAM module, daemon, uid/gid based rules

• Illustrative cgroup use cases
  • Database workload dedicated 90%, background backup utility 10%
  • Virtualized hosting provider – allows QoS (quality of service guarantees based on pricepoint)
CGroup Controllers

- **memory**: Memory controller
  - Allows for setting limits on RAM and swap usage and querying cumulative usage of all processes in the group
- **cpuset**: CPU set controller
  - Binding of processes within a group to a set of CPUs and controlling migration between CPUs
- **cpuacct**: CPU accounting controller
  - Information about CPU usage for a group of processes
- **cpu**: CPU schedular controller
  - Controlling the priorization of processes in the group. Think of it as a more advanced nice level
- **devices**: Devices controller
  - Access control lists on character and block devices
Hierarchy example
Subsystems - memory

- Limit memory usage of **processes** in a group
- Parameters (see memory.txt):
  
  ```
  memory.limit_in_bytes – maximum allowed memory usage by tasks in the group.
  
  memory.max_usage_in_bytes – maximum of used memory.
  
  memory.stat – current memory statistics (RSS, swap, ...)
  ```

- Examples:
  
  - HTTP can take only 30% of memory.
Subsystems - cpuacct

• Computes CPU cycles, burned by members of the group.

• Parameters:

  \texttt{cpuacct.usage} – nr. of cycles.

  \texttt{cpuacct.usage_percpu} – nr. of cycles per CPU.

• Example:

  • Members of 'daemons' used $10^7$ cpu cycles.

  • Out of that, only $2 \times 10^6$ cpu cycles were exhausted by SQL.
Subsystems - cpu

• Set scheduler priority.

• Parameters:
  
  `cpu.shares` – priority of threads in this group, relative to other groups.

• Example:
  
  • SQL can take 2x more CPU cycles than HTTP.
Using CGroups

- Install cgroups support
  - `yum install libcgroup`
  - `apt-get install cgroup-bin libcgroup1`
- Setup a basic `/etc/cgconfig.conf`
  ```
  mount {
    cpuset = /cgroup/cpuset;
    cpu = /cgroup/cpu;
    cpuacct = /cgroup/cpuacct;
    memory = /cgroup/memory;
  }
  ```
- Start the cgroups daemon
  - `service cgconfig start`
Command Line Tools

- **cgexec**
  - Start new process in specified group(s).

- **cgclassify**
  - Move process to specified group(s).

- **cgcreate / cgdelete**
  - Create and remove cgroups manually

- **cgset**
  - Modify defined cgroup
Apache Example

• Edit /etc/cgconfig.conf

```
  . group http {
    .   memory {
    .     memory.limit_in_bytes = 1024M;
    .   }
  .}
```

• Next add this to the /etc/sysconfig/httpd.conf:

```
  . CGROUP_DAEMON="memory:/http"
```

• Then start cgconfig service and httpd
CGroups and Virtual Machines

- Allows to control libvirtd and any other process in the cgroup “virt”
  - Examples are memory ceiling / capping
  - Restrict which CPUs libvirt can utilise
- Add these rules to /etc/cgconfig.conf
  ```
  group virt {
    memory {
      memory.limit_in_bytes = 3.5G;
    }
    cpuset {
      cpuset.cpus = 1-3;
    }
  }
  ```
- Modify /etc/sysconfig/libvirtd and add
  ```
  CGROUP_DAEMON="memory:/virt"
  ```
cgrred

- Daemon, distributes processes to groups according to their user and group id
- Configured by /etc/cgrules.conf:

<table>
<thead>
<tr>
<th>User</th>
<th>Type</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>john</td>
<td>cpu</td>
<td>operators</td>
</tr>
<tr>
<td>@operator</td>
<td>cpu</td>
<td>operators</td>
</tr>
<tr>
<td>apache</td>
<td>cpu</td>
<td>daemons/http</td>
</tr>
<tr>
<td>maria</td>
<td>devices</td>
<td>staff</td>
</tr>
<tr>
<td>maria:ftp</td>
<td>devices</td>
<td>staff/ftp</td>
</tr>
</tbody>
</table>
References

- **RHEL 6 Resource Management Guide**

- **Fedora Overview**
  - [http://fedoraproject.org/wiki/Features/ControlGroups](http://fedoraproject.org/wiki/Features/ControlGroups)

- **Manage Your Performance with Cgroups and Projects**

- **Zonker at ServerWatch on Cgroups**

- **Using Cgroups under Debian**