

Here's a respin of my earlier CPU controller to work on top of Paul Menage's process container patches.

Problem:

Current CPU scheduler is very task centric, which makes it difficult to manage cpu resource consumption of a group of (related) tasks.

For ex: with the current O(1) scheduler, it is possible for a user to monopolize CPU simply by spawning more and more threads, causing DoS to other users.

Requirements:

A few of them are:

- Provide means to group tasks from user-land and specify limits of CPU bandwidth consumption of each group. CPU bandwidth limit is enforced over some suitable time period. For ex: a 40% limit could mean the task group's usage is limited to 4 sec every 10 sec or 24 sec every minute.
- Time period over which bandwidth is controlled to each group to be configurable (?)
- Work conserving - Do not let the CPU be idle if there are runnable tasks (even if that means running task-groups that are above their allowed limit)
- SMP behavior - Limit to be enforced on all CPUs put together
- Real-time tasks - Should be left alone as they are today? i.e real time tasks across groups should be scheduled as if they are in same group
- Should cater to requirements of variety of workload characteristics, including bursty ones (?)

Salient points about this patch:

- Each task-group gets its own runqueue on every cpu.

- In addition, there is an active and expired array of task-groups themselves. Task-groups that have expired their quota are put into expired array.
- Task-groups have priorities. Priority of a task-group is the same as the priority of the highest-priority runnable task it has. This I feel will retain interactiveness of the system as it is today.
- Scheduling the next task involves picking highest priority task-group from active array first and then picking highest-priority task within it. Both steps are O(1).
- Tokens are assigned to task-groups based on their assigned quota. Once they run out of tokens, the task-group is put in an expired array. Array switch happens when active array is empty.
- SMP load-balancing is accomplished on the lines of smpnice.

Results of the patch

=====

Machine : 2way x86_64 Intel Xeon (3.6 GHz) box

Note: All test were forced to run on only one CPU using cpusets

1. Volanomark [1]

 Group A [50% limit] Group B [50% limit]

Elapsed time 35.83 sec 36.6002
 Avg throughput 11179.3 msg/sec 10944.3 msg/sec

 Group A [80% limit] Group B [20% limit]

Elapsed time 23.4466 sec 36.1857
 Avg throughput 17072 msg/sec 11080 msg/sec

2. Kernel compilation

```
Group A [50% limit] Group B [50% limit]
time -p make -j4 bzImage time -p make -j8 bzImage

real 771.00 sec 769.08 sec
```

```
Group A [80% limit] Group B [20% limit]
time -p make -j4 bzImage time -p make -j8 bzImage

real 484.12 sec 769.70 sec
```

--
Regards,
vatsa

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
