Subject: Re: Re: [RFC] Virtualization steps
Posted by Herbert Poetzl on Thu, 13 Apr 2006 01:05:06 GMT
View Forum Message <> Reply to Message

On Wed, Apr 12, 2006 at 12:28:56PM +0400, Kirill Korotaev wrote:
> Sam,
>
> >Ok, I'll call those three VPSes fast, faster and fastest.
> >
> >"fast"   : fill rate 1, interval 3
> >"faster" : fill rate 2, interval 3
> >"fastest" : fill rate 3, interval 3
> >
> >That all adds up to a fill rate of 6 with an interval of 3, but that is
> >right because with two processors you have 2 tokens to allocate per
> >jiffie. Also set the bucket size to something of the order of HZ.
> >
> >You can watch the processes within each vserver's priority jump up and
> >down with `vtop' during testing. Also you should be able to watch the
> >vserver's bucket fill and empty in /proc/virtual/XXX/sched (IIRC)
> >
> >I mentioned this earlier, but for the sake of the archives I'll repeat -
> >if you are running with any of the buckets on empty, the scheduler is
> >imbalanced and therefore not going to provide the exact distribution you
> >asked for.
> >
> >However with a single busy loop in each vserver I'd expect the above to
> >yield roughly 100% for fastest, 66% for faster and 33% for fast, within
> >5 seconds or so of starting those processes (assuming you set a bucket
> >size of HZ).
>
> Sam, what we observe is the situation, when Linux cpu scheduler spreads
> 2 tasks on 1st CPU and 1 task on the 2nd CPU. Std linux scheduler
> doesn't do any rebalancing after that, so no plays with tokens make the
> spread to be 3:2:1, since the lowest priority process gets a full 2nd
> CPU (100% instead of 33% of CPU).
>
> Where is my mistake? Can you provide a configuration where we could test
> or the instuctions on how to avoid this?

well, your mistake seems to be that you probably haven't
tested this yet, because with the following (simple)
setups I seem to get what you consider impossible
(of course, not as precise as your scheduler does it)

```
vcontext --create --xid 100 ./cpuhog -n 1 100 &
vcontext --create --xid 200 ./cpuhog -n 1 200 &
vcontext --create --xid 300 ./cpuhog -n 1 300 &

vsched --xid 100 --fill-rate 1 --interval 6
vsched --xid 200 --fill-rate 2 --interval 6
vsched --xid 300 --fill-rate 3 --interval 6

vattribute --xid 100 --flag sched_hard
vattribute --xid 200 --flag sched_hard
vattribute --xid 300 --flag sched_hard


  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
   39 root      25   0  1304  248  200 R   74  0.1   0:46.16 ./cpuhog -n 1 300
   38 root      25   0  1308  252  200 H   53  0.1   0:34.06 ./cpuhog -n 1 200
   37 root      25   0  1308  252  200 H   28  0.1   0:19.53 ./cpuhog -n 1 100
   46 root       0   0  1804  912  736 R    1  0.4   0:02.14 top -cid 20
```

and here the other way round:

```
vsched --xid 100 --fill-rate 3 --interval 6
vsched --xid 200 --fill-rate 2 --interval 6
vsched --xid 300 --fill-rate 1 --interval 6

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
   36 root      25   0  1304  248  200 R   75  0.1   0:58.41 ./cpuhog -n 1 100
   37 root      25   0  1308  252  200 H   54  0.1   0:42.77 ./cpuhog -n 1 200
   38 root      25   0  1308  252  200 R   29  0.1   0:25.30 ./cpuhog -n 1 300
   45 root       0   0  1804  912  736 R    1  0.4   0:02.26 top -cid 20
```

note that this was done on a virtual dual cpu
machine (QEMU 8.0) with 2.6.16-vs2.1.1-rc16 and
that there were roughly 25% idle time, which I'm
unable to explain atm ...

feel free to jump on that fact, but I consider
it unimportant for now ...

best,
Herbert

> Thanks,
> Kirill