

> currently against an older kernel

ah .. which older kernel?

I tried it against the broken out 2.6.23-rc8-mm2 patch set, inserting it before the task-containersv11-* patches, but that blew up on me - three rejected hunks.

Any chance of getting this against a current cgroup (aka container) kernel?

Could you use the diff --show-c-function option when composing patches - they're easier to read that way - thanks.

```
+ if (!retval) {  
+   cpus_allowed = cpuset_cpus_allowed(p);  
+   if (!cpuset_subset(new_mask, cpus_allowed)) {  
+     /*  
+      * We must have raced with a concurrent cpuset  
+      * update. Just reset the cpus_allowed to the  
+      * cpuset's cpus_allowed  
+      */  
+     new_mask = cpus_allowed;
```

This narrows the race, perhaps sufficiently, but I don't see that it guarantees closure. Memory accesses to two different locations are not guaranteed to be ordered across nodes, as best I recall. The second line above, that rereads the cpuset cpus_allowed, could get an old value, in essence.

```
cpuset update task  sched_setaffinity task  
-----
```

```
A. write cpuset [Q]  V. read cpuset [Q]  
B. read task [P]    W. check ok  
C. write task [P]   X. write task [P]  
   Y. reread cpuset [Q]  
   Z. check ok again
```

Two memory locations:

[P] the cpus_allowed mask in the task_struct of the task doing the sched_setaffinity call.

[Q] the cpus_allowed mask in the cpuset of the cpuset to which the sched_setaffinity task is attached.

Even though, from the perspective of location [P], both B. and C. happened before X., still from the perspective of location [Q] the rereading in Y. could return the value the cpuset cpus_allowed had before the write in A. This could result in a task running with a cpus_allowed that was totally outside its cpusets cpus_allowed.

I will grant that this is a narrow window. I won't lose much sleep over it.

> - uses a priority heap to pick the processes to act on, based on start time

This adds a fair bit of code and complexity, relative to my patch. This I do lose more sleep over. There has to be a compelling reason for doing this.

The point that David raises, regarding the interaction of this with hotplug, seems to be a compelling reason for doing -something-different than my patch proposal.

I don't know yet if it compels us to this much code, however.

Any chance you could provide a patch that works against cgroups?

--

I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

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