

Re: [Lse-tech] [PATCH] cpusets – big numa cpu and memory placement

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-10/0673.html>

From: Paul Jackson (pj_at_sgi.com)

Date: 10/04/04

Date: Sun, 3 Oct 2004 20:33:14 -0700
To: "Martin J. Bligh" <mbligh@aracnet.com>

Martin wrote:

- > *Rebalance!*
- > *Ooooh, CPU 3 over there looks heavily loaded, I'll steal something.*
- > *That one. Try to migrate. Oops, no cpus_allowed bars me.*
- > ...
- > *Humpf. I give up.*
- > *... ad infinitum.*
- >
- > *Desperately boring, and rather ineffective.*

Well ... I don't mind unemployed CPUs being borish. It's not that they have much useful work to do. But if they keep beating down the doors of their neighbors trying to find work, that seems disruptive. Won't CPU 3 in your example waste time and suffer increased lock contention, responding to its deadbeat neighbor?

- > > *Likely your same concerns apply to the task->mems_allowed field that*
- > > *I added, in the same fashion, in my cpuset patch of recent.*
- >
- > *Mmm, I'm less concerned about that one, or at least I can't specifically*
- > *see how it breaks.*

Ray Bryant <raybry@sgi.com> is working this now. There are ways to get memory allocated that hurt on our big boxes – such as blowing out one nodes memory with a disproportionate share of the systems page cache pages, due to problems vaguely like the cpus_allowed ones.

The kernel allocator and numa placement policies don't really integrate mems_allowed into their algorithms, but rather are just whacked upside the head anytime they ask if they can allocate on a non-allowed node. They can end up doing suboptimal placement on big boxes.

A common one is that the first node in a multiple-node cpuset gets a bigger memory load from allocations initiated on nodes up stream of it, that weren't allowed to roost closer to home (or something like this ...

not sure I said this one just right).

Ray is leaning on me to get some kind of memory policy in each cpuset. I'm giving him a hard time back over details of what this policy structure should look like, buying time while I try to make more sense of this all.

I've added him to the cc list here – hopefully he will find my characterization of our discussions amusing ;).

> > *Somewhat like dual-channelled disks, having more than one*
> > *sched_domain apply at the same time to a given CPU leads to confusions*
> > *best avoided unless desperately needed.*
>
> *Agreed. The cpus_allowed mechanism doesn't seem well suited to heavy use*
> *anyway (I think John Hawkes had problems with it too).*

The various problems Hawkes had were various race conditions using the new (at the time) set_cpus_allowed() that Ingo (I believe) added as part of the O(1) scheduler. SGI was on the bleeding edge of using the set_cpus_allowed() call in new and exciting ways, and there were various race and lock conditions and issues with making sure the per-cpu migration threads stayed home.

Other than reminding us that this stuff is hard, these problems Hawkes dealt with don't, to my understanding, shed any light on the new issue uncovered in this thread, that a simple per-task cpus_allowed mask, heavily used to affect affinity policy, can interact poorly with sophisticated schedulers trying to balance an entire system.

===

In sum, I am tending further in the direction of thinking we need to have scheduler and allocation policies handled on a "per-domain" basis, where these domains take the form of a partition of the system into equivalence classes corresponding to subtrees of the cpuset hierarchy.

For example, just to throw out a wild and crazy idea, perhaps instead of one global set of zonelists (one per node, each containing all nodes, sorted in various numa friendly orders), rather there should be a set of zonelists per memory-domain, containing just the nodes therein (subsetted from the global zonelists, preserving order).

We'll have to be careful here. I suspect that the tolerance of those running normal sized systems for this kind of crap will be pretty low.

Moreover, the scheduler in particular, and the allocator somewhat as well, are areas with a long history of intense technical development. Our impact on these areas has to be simplistic, so that folks doing the real work here can keep our multi-domain stuff working with almost no mind to it at all.

Linux-Kernel: Re: [Lse-tech] [PATCH] cpusets – big numa cpu and memory placement

--

I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.650.933.1373

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@vger.kernel.org

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>