

Re: rusage breakdown and cpu limits.

Re: rusage breakdown and cpu limits.

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2007-05/msg00194.html>

- *From:* Bruce Evans <brde@xxxxxxxxxxxxxxxx>
 - *Date:* Wed, 30 May 2007 07:10:23 +1000 (EST)
-

On Tue, 29 May 2007, Jeff Roberson wrote:

The problem with using a pool or per-process spinlock is that it keeps the contention in the process domain, rather than thread domain. For multithreaded processes this will give the same contention as a global scheduler lock, only slightly reduced in scope. I'd like to solve this in such a way that we don't have to revisit it again.

I think I'm going to make the rusage struct per-thread and aggregate it on demand. There will be a lot of code churn, but it will be simple. There are

Ugh.

a few cases where which will be complicated, and cpulimit is one of them.

No, cpulimit is simple because it can be fuzzy, unlike `calcru()` which require the rusage to be up to date.

I see how rusage accumulation can help for everything *_except_* the runtime and tick counts (i.e., for stuff updated by `statclock()`). For the runtime and tick counts, the possible savings seem to be small and negative. `calcru()` would have to run the accumulation code and the accumulation code would have to acquire something like `sched_lock` to transfer the per-thread data (since the lock for updating that data is something like `sched_lock`). This has the same locking overheads and larger non-locking overheads than accumulating the runtime directly into the rusage at context switch time -- `calcru()` needs to acquire something like `sched_lock` either way.

Bruce

freebsd-arch@xxxxxxxxxxxxx mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-arch>

To unsubscribe, send any mail to "freebsd-arch-unsubscribe@xxxxxxxxxxxxx"

Re: rusage breakdown and cpu limits.