

Re: Updated rusage patch

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2007-06/msg00041.html>

- *From:* Attilio Rao <attilio@xxxxxxxxxxx>
 - *Date:* Fri, 08 Jun 2007 16:10:05 +0200
-

Jeff Roberson wrote:

On Wed, 6 Jun 2007, Jeff Roberson wrote:

On Tue, 5 Jun 2007, Bruce Evans wrote:

This can probably be fixed more simply by calling `rufetch()` to reset the time state in threads as a side effect. Do this before resetting the state in the process.

Ok, I agree with bde here, just call `rufetch` and this will clear each thread, and then you can clear the `rux` in the `proc`.

I'd like to make a list of the remaining problems with `rusage` and potential fixes. Then we can decide which ones myself and `attilio` will resolve immediately to clean up some of the effect of the `sched` lock changes.

1) The `ruadd()` in `thread_exit()` is not safe since we're accessing another thread's unlocked `rusage` structure. Potential solution is to allocate `p_ru` as part of the `proc` struct and add into there, which will be protected by the `PROC_SLOCK`, which bde seemed to like better anyway.

2) We may lose information between `exit1()` and `thread_exit()` due to the way `p_ru` is initialized before we're done exiting. There also seems to be a race where `wait()` operates on a process before it's done in `thread_exit()` which means `wait` may return `rusage` information without the child added in! The solution will be to fix this race, and then access `p_ru` directly in `wait()`.

The patch at <http://people.freebsd.org/~jeff/rusage3.diff> fixes points 1 and 2 as well as the `p_runtime` initialization problem. This moves the collection of child `rusage` back into `exit1()` and changes the exiting threads to accumulate their `rusage` into `p_ru` under protection of the process spinlock. This also removes the gross lock/unlock of `proc` `slock` (formerly

Re: Updated rusage patch

sched_lock) from wait and implements something more sensible.

Jeff

3) There is no locking around rufetch() and calcru(). calcru() may apply new rux values to an old rusage, giving inaccurate results. The solution is to either require the proc slock around both calls, or provide a new routine which does the fetch and calc while grabbing the lock itself.

And this should fix (3):

<http://users.gufi.org/~rookie/works/patches/schedlock/rusage2.diff>

(and reorders rucollect() declaration sorted by name).

A thought:

Shouldn't we actually remove in calcru() (and rufetchcalc()) the copy to the rux object?

When sched_lock was there it would be useful since it had a lot of contention, but now that the per-proc spinlock is protecting those fields it is useless. And consider that calcru1() has no locking inside (so you won't expect particularly long execution times).

Thanks,

Attilio

freebsd-arch@xxxxxxxxxxxxx mailing list

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

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