

Re: Comments on the KSE option

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/current/2006-10/msg00559.html>

- *From:* Robert Watson <rwatson@xxxxxxxxxxx>
 - *Date:* Sat, 28 Oct 2006 20:47:05 +0100 (BST)
-

On Sat, 28 Oct 2006, Paul Allen wrote:

From Robert Watson <rwatson@xxxxxxxxxxx>, Sat, Oct 28, 2006 at 11:04:48AM +0100:

This is my single biggest concern: our scheduling, thread/process, and context management paths in the kernel are currently extremely complex. This has a number of impacts: it makes it extremely hard to read and understand, it adds significant overhead, and it makes it quite hard to modify and optimize for increasing numbers of processors. We need to be planning on a world of 128 hardware threads/machine on commodity server hardware in the immediate future, which means that the current "giant sched_lock" cannot continue much longer. Kip's prototypes of breaking out sched_lock as part of the sun4v work have been able to benefit significantly from the reduced complexity of a KSE-free kernel, and it's fairly clear that the task of improving schedule scalability is dramatically simpler when the kernel model for threading is more simple. Regardless of where the specific NO_KSE option in the kernel goes, reducing kernel scheduler/etc complexity should be a first order of business, because effective SMP work really depends on that happening.

Let us suppose that this M:N business is important, perhaps something to consider is why and whether the kernel has so much knowledge of it.

If I read Matt Dillon's comment closely enough, I believe his precise recommendation was not "something like kse as Julian read it" but rather something where this M:N component was entirely part of the userland threading support and therefore would just go away or not depending on which library you linked with.

I think posix might require a global priority space though...

Anyways it remains dubious in my mind that the kernel should allow a user to create many processes but penalize creating threads.

The only reason I can think of is that you expect people to be sloppy with their threads and careful with their processes.

Re: Comments on the KSE option

Still if I am ray-tracing why should I need to make a point of picking my thread/process balance to get around your mechanism. If fairness is the goal why am I even allowed to do so?

I think the notion of fairness is orthogonal to M:N threading. M:N is about efficiently representing user threading to kernel space, as well as avoiding kernel involvement in user context switches when not needed. Fairness is about how the kernel allocates time slices to user processes/threads. Fairness can be implemented for both 1:1 and M:N, with the primary differences being in bookkeeping.

Programmers often use threading based on a misunderstanding about performance. Threading actually increases kernel lock contention when compared with using processes for parallelism, so if the benefits from address space sharing don't outweigh the added costs of contention, threaded applications can run significantly slower than multi-process ones. Many programmers believe that threading is necessarily faster than using multiple processors, so I think we're fundamentally forced to deal with the way they do use them, rather than how they should use them.

Robert N M Watson
Computer Laboratory
University of Cambridge

freebsd-current@xxxxxxxxxxx mailing list
<http://lists.freebsd.org/mailman/listinfo/freebsd-current>

To unsubscribe, send any mail to "freebsd-current-unsubscribe@xxxxxxxxxxx"