

Re: Network Stack Locking

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2004-05/0103.html>

From: Robert Watson (rwatson_at_freebsd.org)

Date: 05/26/04

Date: Wed, 26 May 2004 17:26:31 -0400 (EDT)
To: Matthew Dillon <dillon@apollo.backplane.com>

On Tue, 25 May 2004, Matthew Dillon wrote:

> *This sounds very similar to an interrupt mechanism I designed about
> a decade ago. Instead of saving and restoring the interrupt
> context for the interrupt thread, the thread switch was special-cased
> to basically create a context (by pushing a procedure call on the stack)
> on switch-in and to throw it away on switch-out (which was always at
> the end of the interrupt routine). I extended the same mechanism down
> into 'userland' by creating a 'waitforever()' system call which basically
> did nothing but wait for and dispatch signal vectors (most of the programs
> were event oriented and had no main loop). The nice thing about this was
> that no context had to be saved while the program was sitting in
> waitforever(), or restored when the program returned from a signal handler.
> This more than doubled scheduler performance (which on a 10MHz 68000 was
> important).*
>
> *In many ways, the continuation mechanism and the message queue mechanism
> appear to be nearly identical. If an explicit exit from a procedure
> is required to optimize the stack with the continuation mechanism, then
> that isn't much different than moving the message to an event queue
> and returning to the message processing loop. Neither case allows
> you to save stack context or to save the current procedural stacking
> level, and both mechanisms allow you to reuse your current stack to
> handle multiple messages/continuations.*

I'm not sure if you've spent much time looking at the Mach kernel and literature, but a lot of the concepts in AmigaOS and DFBSD are highly parallel to concepts in Mach (and hence in many derived systems). The original Mach project at CMU (from about 1985-1993) still has a web page:

<http://www-2.cs.cmu.edu/afs/cs/project/mach/public/www/mach.html>

I don't remember if there's specifically a paper on continuations there, but it should be discussed in their developer guides, threading and parallelism bits, and documentation of their scheduler.

freebsd-arch: Re: Network Stack Locking

There's also been follow-up work on Mach in a great many environments, including work at OSF, University of Utah (FLUX, etc), by the GNU folks on Hurd, NeXT, Apple, et al. There's also been follow-up work at CMU on real time scheduling in Mach. The reason I tend to raise aspects of Darwin when responding to your e-mails regarding DFBSD is that, while the details are often pretty different, the general approach reveals many parallels.

Robert N M Watson FreeBSD Core Team, TrustedBSD Projects
robert@fledge.watson.org Senior Research Scientist, McAfee Research

freebsd-arch@freebsd.org mailing list

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

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