

Special schedulers, one CPU only kernel, one only userland

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2005-08/0111.html>

From: Andre Oppermann (andre_at_freebsd.org)

Date: 08/10/05

Date: Wed, 10 Aug 2005 14:02:58 +0200

To: arch@freebsd.org

When using FreeBSD as a high performance router there are some desirable changes to the way multiple CPUs are handled. Normally a second CPU doesn't add much (if any) performance to routing because of locking overhead and packets randomly being processed by the CPUs wasting cache efficiency. On the other hand having just one CPU is not optimal in running the routing daemon in userland. When there are large changes to the table (eg. BGP full feed flap) userland sucks time away from the packet forwarding in the kernel.

The idea is to combine both worlds by designating CPU0 exclusively for all kernel processing (thus avoiding the expensive mutex synchronization and bus locking instructions) and CPU1 exclusively for all userland processing. Whenever a userland program does a syscall the kernel CPU will take over. When it's done, the process get run by the userland CPU again. That way we get a very good scalability out of two CPUs for this particular task.

Hence my question to the SMP and scheduler gurus: How well does the current SMP and scheduler architecture lend itself to this kind of special handling? Is it just a matter of modifying (or plugging in) the schedule or are there more involved things to consider?

--

Andre

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"