

Re: How to develop a random number generation device

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2007-09/msg03197.html>

- *From:* David Brown <david.brown@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 17 Sep 2007 18:40:35 +0200
-

John Larkin wrote:

On Sun, 16 Sep 2007 22:07:42 +0200, David Brown
<david.brown@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

John Larkin wrote:

On Sun, 16 Sep 2007 11:33:21 -0700, MooseFET
<kensmith@xxxxxxxxx>
wrote:

On Sep 15, 11:09 am, John Larkin
<jjlar...@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
wrote:
[...]

architecture. In a few years
we'll have, say, 1024
processors on a
chip, and something new
will be required to manage
them. It will be a
thousand times simpler and
more reliable than
Windows.

I think that the number of virtual cores will
grow faster than the
number fo real cores. With extra register
banks and a bit of clever
design, a single ALU can look like two
slightly slower ones.

I expect to see multicore machines with less
actual floating point
ALUs than actual integer ALUs.

Re: How to develop a random number generation device

Sounds sort of like Sun's Niagra chips, which have (IIRC) 8 cores, each with 4 threads, but only a few floating point units. For things like web serving, it's ideal.

Yup. Low-horsepower tasks can just be a thread on a multithread core, and many little tasks don't need a dedicated floating-point unit.

My point/fantasy is that OS design should change radically if many, many real or virtual CPUs are available. One CPU would be the manager, and every task, process, or driver could have its own, totally confined and protected, CPU, and there would be no context switching ever, and few interrupts in fact.

That's not going to work for Linux, anyway – there is a utility thread spawned per cpu at the moment (work is underway to avoid this, because it is a bit of a pain when you have thousands of cpus in one box).

However, there is no point in having a cpu (or even a virtual cpu) dedicated to each task. Many sorts of tasks spend a lot of time sleeping while waiting for other events – a cpu in this state is a waste of resources.

Only if you think of a CPU as a valuable resource. As silicon shrinks, a CPU becomes a minor bit of real estate. It makes sense to use it when there's something to do, and put it to sleep when there's not. Lots of power gets saved by not doing context switches.

CPUs *are* a valuable resource – modern cpu cores take up a lot of space, even when you exclude things like the cache (which take more space, but cost less per mm² since you can design in a bit of redundancy and thus tolerate some faults).

The more CPUs you have, the more time and space it costs to keep caches and memory accesses coherent. There are some sorts of architectures which work well with multiple CPU cores, but these are not suitable for general purpose computing.

My point is that large numbers of CPU cores *will* become common and cheap, and we need a new type of OS to take advantage of this new reality. Done right, it could be simple and astoundingly secure and reliable.

Re: How to develop a random number generation device

I would be very surprised to see a system where the number of CPU cores was greater than the number of processes. I expect to see the number of cores increase, especially for server systems, but I don't expect to see systems where it is planned and expected that most cores will sleep most of the time.

I'd be happy to waste a little silicon if I could have an OS that doesn't crash and that doesn't go to sleep for seconds at a time for no obvious reason.

Multiple cores gives absolutely no benefits in terms of reliability or stability – indeed, it opens all sorts of possibilities for hard-to-debug race conditions.

.