

Re: How to develop a random number generation device

[...]

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

Re: How to develop a random number generation device

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.

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.

Re: How to develop a random number generation device

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.

Re: How to develop a random number generation device

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

Re: How to develop a random number generation device

reality. Done right, it could be simple and astoundingly secure and reliable.

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.

Well, I remember 64-bit static rams, and 256-bit DRAMS. I can't see any reason we couldn't have 256 or 1024 cpu's on a chip, especially if a lot of them are simple integer RISC machines.

You can certainly get 1024 CPUs on a chip – there are chips available today with hundreds of cores. But there are big questions about what you can do with such a device – they are specialised systems. To make use of something like that – you'd need a highly parallel problem (most desktop applications have trouble making good use of two cores – and it takes a really big web site or mail gateway to scale well beyond about 16 cores). You also have to consider the bandwidth to feed these cores, and be careful that there are no memory conflicts (since cache coherency does not scale well enough).

No, no, NO. You seem to be assuming that we'd use multiple cores the way Windows would use multiple cores. I'm not talking about solving big math problems; I'm talking about assigning one core to be a disk controller, one to do an Ethernet/stack interface, one to be a printer driver, one to be the GUI, one to run each user application, and one to be the system manager, the true tiny kernal and nothing else. Everything is dynamically loadable, unloadable, and restartable. If a core is underemployed, it sleeps or runs slower; who cares if transistors are wasted? This would not be a specialized system, it would be a perfectly general OS with applications, but no process would hog the machine, no process could crash anything else, and it would be fundamentally reliable.

This is not about performance; hardly anybody needs gigaflops. It's all about reliability.

Re: How to develop a random number generation device

Re: How to develop a random number generation device

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.

They don't if you insist on running a copy of a bloated OS on each. A system designed, from scratch, to run on a pool of cheap CPUs could be incredibly reliable.

That's a conjecture plucked out of thin air. Of course a dedicated OS designed to be limited but highly reliable is going to be more reliable than a large general-purpose OS that must run on all hardware and support all sorts of software – but that has absolutely nothing to do with the number of cores!

Programmers have pretty much proven that they cannot write bug-free large systems. Unless there's some serious breakthrough – which is really prohibited by the culture – the answer is to have the hardware, which people *do* routinely get right, take over most of the functions that an OS now performs. One simple way to do that is to have a CPU per process. It's going to happen.

When I was just a sprout, my old mentor Melvin Goldstein told me "in these integrated circuit things, one day transistors could cost a penny each." I thought he was crazy. OK, one day CPUs will cost 5 cents each, and Windows is not the ultimate destiny of computing.

Hey, he wrote a book!

<http://www.amazon.com/Physics-Foibles-physics-computer-students/dp/1553957768>

John

.

Re: How to develop a random number generation device