Re: a dozen cpu's on a chip

Source: http://sci.tech−archive.net/Archive/sci.electronics.design/2008−05/msg01898.html

- *From*: John Larkin <jjlarkin@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
- *Date*: Fri, 16 May 2008 08:14:25 −0700

On Fri, 16 May 2008 11:44:47 +0100, Martin Brown
<|||newspam|||@nezumi.demon.co.uk> wrote:

> John Larkin wrote:
>
> > On Thu, 15 May 2008 10:24:05 −0700 (PDT), panteltje@xxxxxxxxx wrote:
> >
> > > On 15 mei, 15:50, John Larkin
> > > <jjlar...@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:
> > >
> > > > On Thu, 15 May 2008 09:14:43 +0100, John
> > > > Devereux
> > > > Nanometer transistors are fast and free.
> > >
> > > Actually they are not, those 80−cores will be difficult to
> > > make
> > > (yield),
> >
> > If you want 250 cores, build 300 and use the 250 that work. So a chip
> > can have 50 defects and you can still sell it. Or build one giant CPU
> > on the same silicon and toss it if has a single defect.
>
> Unless and until there is software to efficiently exploit large
> processor clusters for general purpose use it doesn't matter.
>
> > > > gigabit world. The majority of guys here are
> > > > adamant that
> > > > things will never change, a pretty radical
> > > > position for engineers to
> > > > take.
> > >
> > > mm you keep sticking that in every bodies mouth, but when I

> > asked how
> > you would spread a monolithic resources sucking application
> > over 'n'
> > CPUs
> > you remained silent.
>
> I already suggested that a few of the cpu's could be floating−point monster number crunchers, and most could be dumber, slower integer machines. A TCP/IP stack doesn't need much floating point power.

Neither does the core kernel for an operating system. Your model serves only to waste silicon real estate and electrical power to no good end.

> > And that is one issue.
> > The other one you conveniently forget is that, if each core
> > has its
> > own memory,
> > where is the overhead in moving data... sync. etc.
>
> They'd surround a shared cache. They wouldn't bother the common cache when they execute out of local cache, or when the use the small local stack and variables rams. That makes the shared cache much more efficient, since it not being invalidated by a lot of unnecessary traffic.

Want to bet?

The fastest way to bring your "uncrashable" independent CPUs with shared common memory model to its knees would be to set a few small tasks running flat out in several cores allocating and deallocating memory at random and hiting it with read/writes at worst case strides for the cache. The OS would still run but its performance would be dire.

Then don't let that happen. That part is easy and obvious.

> One thing I've always thought that CPUs should have is hardware task switching, a register that declares which task or thread the core is running. That would instantly remap everything... the registers, the memory mapping, everything. That would make context switching have

> > zero overhead, and allow full hardware protection. But nowadays, one
>
> There are CPUs coming along (in production?) with hardware support for threads and context switching. And that does make good sense. Some extra hardware support for memory allocation and garbage collection might also be handy but is not mainstream.

Of course it's not mainstream. The question is whether "mainstream" will ever change. A lot of people are arguing that it never will. I suppose there were people who thought that console hi−fi and black−and−white TVs and portable manual typewriters were the ultimate in home automation.

John

> > might just as well have multiple cores. That would be faster, and avoids some cache efficiency and pipeline issues.
>
> But create all sorts of other I/O bandwidth bottlenecks that you conveniently gloss over in your hazy rose tinted view.

How can making the whole system more efficient, reducing cache and main memory traffic, and eliminating context switching cause bottlenecks?

> > It is interesting that 100% of the responses to my posts have been destructive, and none additive. I sure hope you guys don't actually work that way.
>
> That is because your idea would not work as you intend and you are completely deaf to any criticism.

All I hear is criticism; nobody picks up on the fact that the chip manufacturers *are* building or planning 32 and 64 core processors, and that *might* really change the way OS's are designed.

What's really interesting here isn't the technology, it's the

psychology.

> The research work at Intel is on speculative multi−threading and other methods to allow multicore hardware to deliver real world performance increases in the future − a short review online at:
>
> http://www.intel.com/technology/magazine/research/speculative−threading−1205.htm

Intel has an impressive record of a) investing in the status quo and b) wildly missing the mark on everything else. They sure have an engineering mentality!

John

.