

Re: Intel details future Larrabee graphics chip

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2008-08/msg01470.html>

- *From:* John Larkin <jjlarkin@xx>
 - *Date:* Fri, 08 Aug 2008 07:57:48 -0700
-

On Fri, 08 Aug 2008 10:03:28 +0100, Martin Brown
<||||newspam|||@nezumi.demon.co.uk> wrote:

John Larkin wrote:

On Thu, 07 Aug 2008 14:51:57 GMT, Jan Panteltje
<pNaonStpealmtje@xxxxxxxx> wrote:

On a sunny day (Thu, 07 Aug 2008 07:08:52 -0700) it
happened John Larkin
<jjlarkin@xx> wrote
in
<d10m94d7etb6sfcm3hmdl3hk8qnels3kg@xxxxxxxx>:

Been there – done that :-)

That is precisely how the
early SMP systems worked,
and it works
for dinky little SMP systems
of 4–8 cores. But the kernel
becomes
the bottleneck for many
workloads even on those,
and it doesn't
scale to large numbers of
cores. So you HAVE to
multi-thread the
kernel.

Why? All it has to do is grant run
permissions and look at the big
picture. It certainly wouldn't do I/O or
networking or file

Re: Intel details future Larrabee graphics chip

management. If memory allocation becomes a burden, it can set up four (or fourteen) memory-allocation cores and let them do the crunching.

Why multi-thread *anything* when hundreds or thousands of CPUs are available?

Using multicore properly will require undoing about 60 years of thinking, 60 years of believing that CPUs are expensive.

Thinking multicore properly might yield some advantages on certain types of problem. But these are not the sort of problems most domestic users of PCs actually have. It could be useful for 3D gaming, but even there it still makes sense to split the load across specialised dedicated video CPUs using fancy memory and generics doing the grunt work.

Ah, and this all reminds me about when 'object oriented programming' was going to change everything.

It did lead to such language disasters as C++ (and of course MS went for it),

where the compiler writers at one time did not even know how to implement things.

Now the next big thing is 'think an object for every core' LOL.

Days of future wasted.

All the little things have to communicate and deliver data at the right time to the right place.

Sounds a bit like Intel made a bigger version of Cell.

And Cell is a beast to program (for optimum speed).

Then stop thinking about optimum speed. Start thinking about a computer system that doesn't crash, can't get viruses or trojans, is easy to understand and use, that not even a rogue device driver can bring down.

How exactly does your wonder architecture prevent the muppet at the keyboard clicking on the canonical Trojan that starts two new threads and grabs IO and memory resources at random?

The browser runs on one or more CPUs that are totally sandboxed: they are limited in what files they can access, how much memory they can use, everything. Those cpu's can even crash and not take down anything

Re: Intel details future Larrabee graphics chip

but the browser. The tcp/ip, the file system, the user graphics, the disk i/o... all are separate, supervised, trusted processes, each with its own cpu.

Oh it dies when the number of processes running $N > 1024$ (about 10x start process latency after he hits return). Fabulous!

Oh, one *would* have to assume that the system was not coded by morons. I forgot to specify that.

The only way your idea will satisfy your idealised goals is if it remains in fantasy land. Once you apply power it is vulnerable to all the usual human factors Trojan attacks no matter how robust the hardware.

Our FPGA's have dozens of times the number crunching power of a Core Duo, and never get trojans, memory leaks, any of that.

Think about how to manage a chip with 1024 CPUs. Hurry, because it will be reality soon. We have two choices: make existing OS's unspeakably more tangled, or start over and do something simple.

There are tight robust OSs but Windows is not one of them. On the desktop Linux is a lot closer, and the Apples OS X is pretty good too. And IBM's ill fated OS/2 was good in its day (sadly 'Doze won). The PC world could easily have been different had technological superiority won the day (instead of glizty GUI froth and BSODs).

When you have 1024 CPUs you have 1024x1023 (virtual) communications paths to other CPUs. Unless you are very careful it is easy to end up with interprocess communications that are like wading through treacle.

Any OS has a zillion inter-process paths. The difference would be that they are designed by engineers, who can manage things like this, instead of programmers, who can't.

Re: Intel details future Larrabee graphics chip

You might find the following research article interesting.

<http://www2.lifl.fr/west/publi/MDL+07frontiers.pdf>

Connexion machine at MIT goes back to the early 80's and was influenced by Lisp in its early incarnations.

http://en.wikipedia.org/wiki/Connection_Machine

Speed will be a side effect, almost by accident.

If you really think large numbers of CPUs will give some advantage in home computers why not write a program to emulate them using the latest Pentium virtualisation instructions. There may already be an academic implementation of this model – I haven't looked. It would be useful for teaching purposes if nothing else. And it might help disabuse you of some of your wilder notions on this subject.

Sorry, I have a day job.

So what do you think OS's will look like 10 years from now, when even home computers run on chips with 100's of cores? Still one gigantic VMS/Mach/NT descendent running on one CPU, thrashing and piping all over the place, doing everything, still vulnerable to viruses and application bugs, still mixing scheduling and virtual memory management and file systems and running PowerPoint with serial port interrupts? And those other cores stay idle unless you play a game?

Things will never change? We'll always use 1980's OS architectures?

John

.