

Re: Next Generation of Language

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2007-01/msg00514.html>

- *From:* George Neuner <gneuner2/@comcast.net>
 - *Date:* Fri, 12 Jan 2007 17:17:45 -0500
-

On 12 Jan 2007 01:35:29 -0800, "Tim Bradshaw" <tfb+google@xxxxxxx> wrote:

George Neuner wrote:

Well, on Cells the private memories are not cache but staging memories ... the main processor has to move data into and out of them on behalf of the coprocessors.

It doesn't matter very much who moves the data, it's still cache :-). The issue that counts, really, is what the programming model is at the user level. No one should need to care whether things are done automatically by the hardware as most L1/L2 caches are today, or by hardware with substantial SW support as, say, MMUs, or almost entirely by SW with some small amount of HW support, as, say disk paging. (Actually, the second thing that counts is whether the HW can efficiently support the programming model you choose.)

I have considerable experience with manual staging (on DSPs) and I can tell you that it is a royal PITA to schedule several functional units and keep them going full blast using software alone.

Cell is less onerous only because of the granularity of the code the coprocessors can execute – whole functions or miniprograms rather than the baby steps DSP units can take.

AFAIK, no one has tried to offer a hardware solution to staging computations in a distributed memory system since the KSR1 (circa 1990, which failed due to the company's creative bookkeeping rather than the machine's technology). Everyone now relies on software approaches like MPI and PVM.

Re: Next Generation of Language

Well, I think they have actually, in all but name: that's essentially what NUMA machines are. Such machines are quite common, of course (well, for bigger systems anyway): all Sun's recent larger machines (4 & 5-digit sunfire boxes) are basically NUMA, and it may be that smaller ones are too.

Non Uniform Memory Access simply means different memories have different access times – that describes just about every machine made today. The NUMA model distinguishes between "near" and "far" memories in terms of access time, but does not distinguish by how the memories are connected – a system with fast cache and slower main memory fits the model just as well as one with a butterfly network between CPU and memory.

Of course, as I said above, this comes down to programming model and how much HW support you need for it. I think the experience of the last 10–20 years is that a shared memory model (perhaps "shared address space"?), preferably with cache-coherency, is a substantially easier thing to program for than a distributed memory model. Whether that will persist, who knows (I suspect it will, for a surprisingly long time). Of course the physical memory that underlies this model will become increasingly distributed, as it already has to a great extent.

It's all about the programming model and I think you are on the right track. Shared address space is the right approach, IMO, but further I believe it should be implemented in hardware.

That is why I mentioned KSR1 – the only massive multiprocessor I know of that tried to help the programmer. KSR1 was a distributed memory multiprocessor (256..1088 CPUs) with a multilevel caching tree network which provided the programmer with the illusion of a shared memory. The KSR1 ran a version of OSF/1, so software written for any shared memory Unix multiprocessor was relatively easy to port – an important consideration because most people looking to buy a supercomputer were outgrowing a shared memory machine.

There was, of course, a penalty paid for the illusion of shared memory. Estimates were that the cache consistency model slowed the machine by 15–25% vs comparable MPI designs, but IMO that was more than made up for by the ease of programming. The second generation KSR2 improved shared memory speeds considerably, but few people ever saw one – the company went belly up before it was formally introduced.

George

—

for email reply remove "/" from address

Re: Next Generation of Language

Re: Next Generation of Language