Panel: Wall ahead in multicore programming

Rick Merritt

5/3/2011 8:17 PM EDT

SAN JOSE, Calif. – The wall on the horizon in multicore programming is getting closer, according to a panel of experts at the **Multicore Expo** here. Without new tools and methods, programmers will not be able to continue to reap significant benefits from tomorrow's increasingly parallel chips.

Processor designers switched from pushing frequency to delivering multicore designs to avoid hardware problems of power leakage. But that has created an increasingly pressing need for parallel programming tools and methods that don't yet exist.

"The wall is there," said Alex Bachmutsky, a chief architect at Nokia Siemens Networks. "We probably won't have any more products without multicore processors [but] we see a lot of problems in parallel programming," he said.

Alex Bachmutsky

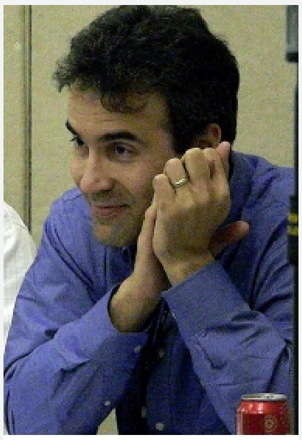
"One of our problems is how to parallelize existing programs with eight or 10 million lines of code--you can rewrite everything but it will cost a lot of money," Bachmutsky said.

"The other problem is we have some algorithms that from their definition were not parallel like Viterbi decoding," he said. "We can change it, but we'd have to change all the cell towers and phones, too, and it's not do-able," he added.

An audience member noted another looming problem: developers can no longer expect next-generation processors will boost performance of their apps. "For 40 years we got a free lunch, but now I wonder what I can write in my code today that will still work in 20 years," he said.

"You can double cores and actually decrease performance," said Rob Muñoz, a distinguished engineer at LSI.

"Parallel software is hard to develop, maintain and evolve," Muñoz said. "There's a graveyard of companies who have tried, but the industry is stuck with multicore because it's the only way to scale chips," he said.


Rob Munoz

LSI has taken a conservative approach to using multicore in its comms chips to limit the software problems, he said.

"A lot of the problems we are trying to address are ones we have had for 30 years in high performance computing, and we haven't solved them yet," said Mike Anderson, chief scientist, with The PTR Group, a consulting firm.

One nasty problem is how to manage multithreaded apps where threads may move between different cores, he said.

"You have to change the way you think about the problem," Anderson said. "People in DSP understand how parallel tasks such as multiply-accumulates and fast Fourier Transforms work, but that's not the majority of developers in the market now," he said.

Asked if the industry needs a new programming language, one panelist said we already have too many of them. "First need to understand what it means to be parallel," he said.

Navigate to related information

Most Popular
1 Analog expert Bob Pease dies in accident
2 Photo gallery: Remembering Jim Williams
3 Electronics and the environment: Five technologies to watch
4 Jim Williams, analog circuit guru, dies
5 Japan takes Sparc to 8 Petaflops
6 Analyst: Yield drove Globalfoundries change
7 Intel-Micron JV to expand Utah fab
8 TI rolls out SoCs for small cell base stations
9 Moore's Law, the bifurcation of the semiconductor industry and 3-D integration
10 OEMs show systems with Intel MIC chips

Comments

Mapou

5/3/2011 10:40 PM EDT

The multicore industry is in deep trouble and, as we know, they've known that this problem would mushroom into a major crisis for at least a decade. The people that are the most to blame for this sad state of affair are none other than the people whom we have entrusted to finding a solution: computer scientists. Why do I say this? Because the reason that we have a crisis is that the baby boomer geeks, who are still in charge of computer science, worship the Turing Computing Model as God's gift to humanity. But guess what? The TCM is the problem, not the solution. It is time for the old boomers to gracefully retire and let a new generation have their turn at the wheel. Sorry, you blew it, guys. Big Time.

It gets worse. Sure, the programming wall is breachable in principle; after all, the most complex computing system known, the brain, is inherently parallel. And the brain accomplishes its magic with extremely slow elementary processors and a very small energy footprint. There are several lessons to be learned here, I'm sure. However, there is another more formidable wall on the horizon and we are approaching it at high speeds. It's called the memory bandwidth problem: As the number of cores continue to increase, the memory bus bottleneck becomes intractable and destroys the performance benefits of having multiple cores.

.

I've been saying this for at least as long as Rick Merritt has been covering the multicore programming crisis. My advice to the industry is this. Realize that our current thread-based, Turing-inspired parallel computing model sucks. I know it and you know it. It's time for a radical change. The more you procrastinate, the worse it's going to get. Google (or Bing) "How to Solve the Parallel Programming Crisis" and do the right thing.

.

And please, stop wasting your money on the Turing worshipers and boomer geeks in general. It has not worked in all the years you've been doing it and it's not about to work any time soon. Telling it like I see it.

Sign in to Reply

resistion

5/4/2011 12:17 AM EDT

The memory bandwidth problem cited by Mapou is actually an interconnect problem. But shortening the interconnect distance would lead to heat dissipation problems. Perhaps embedded DRAM would be more helpful?

Sign in to Reply

greenpattern

5/4/2011 1:39 AM EDT

Yes TSVs would block heat sink with heat source. Embedded DRAM or anything else would share heat sink, avoid this issue.

Sign in to Reply

Mapou

5/4/2011 2:49 AM EDT

The problem is not really the data transfer speed of the connections. We know that there is an upper limit to speed dictated by physics. So there not much we can do about that. The problem has to do with bus contention, i.e., having multiple cores trying to access memory via a single bus at the same time.

Even the slowest connection speeds would not be so bad if we had a memory system that allowed tens or even thousands of cores to have simultaneous random read/write access to multiple data. This is an area that is crying for a breakthrough.

Sign in to Reply

rick.merritt

5/4/2011 2:00 AM EDT

We still have to solve the parallel programming problem--or face pretty big consequences in years to come.

Sign in to Reply

Mapou

5/4/2011 3:50 AM EDT

Absolutely.

Sign in to Reply

GREAT-Terry

5/4/2011 12:24 PM EDT

I don't think 2 cores really give 2 times performance. Parallel programming is a monster that I don't think we have good way to deal with yet. To most products, multi-core is more or less a marketing tactic rather than a real technical advantage!

Sign in to Reply

dougwithau

5/4/2011 12:57 PM EDT

The problem is not technical. It is a mental block provided by the panelists themselves. They even say it right in the article,
"One of our problems is how to parallelize existing programs with eight or 10 million lines of code--you can rewrite everything but it will cost a lot of money," Bachmutsky said.

Sorry boys, but a rewrite is required. The real and true problem of multicore is NOT that the programming is impossible, or even hard. The problem is you can not throw the old crap on the new hardware and call it done. Everyone is searching for the silver bullet that lets us put the poor code, which has been tested into working, onto a multiprocessor system. There are no silver bullets.

Yes, this is costly. Yes this is difficult and error prone. Yes, you should have started the project 2 years ago. Better get started now!

Sign in to Reply

dirk.bruere

5/4/2011 2:53 PM EDT

I was reading articles like this in the 70s, and alternatives being proposed ranged from content addressable memory to dataflow architectures and on. The answer is probably sitting in the IEEE archives somewhere, waiting to be re-invented.

Sign in to Reply

Eric Verhulst_Altreonic

5/4/2011 3:16 PM EDT

The solution is a lot closer than most people think. It looks like a lot of knowledge was lost in the last decade.
See http://www.electronics-eetimes.com/en/News/full-news.html?cmp_id=7&news_id=222906867
(Title: Traditional Programming has hit the power wall).

Sign in to Reply

Code Monkey

5/4/2011 3:18 PM EDT

Manual rewrites are expensive. Why can't they be automated or at least semi-automated. It's a matter of refactoring, with tools finding dependencies among modules and suggesting ways to refactor in a parallel-friendly way. A tool should be able to find and point out the dependency that keeps tasks A and B from running in parallel.

Sign in to Reply

pbinCA

5/6/2011 12:30 PM EDT

Any mapping onto parallel architecture (if it is to accomplish any speedup) MUST understand the run-time flow of information processing. Analyzing source code just won't cut it. If the system processes an external datastream, it gets even more hairy understanding how a parallel processing system reacts in realtime. To do what you are suggesting (automated parallelization as a general-purpose tool) would require that the system purpose or intention be captured in the source code. Clearly, system purpose resides in its human designers, and only gets implemented in the code. This implies that human designers are needed to parallelize an arbitrary serial process (and have it work faster).

Sign in to Reply

fdunn

5/4/2011 4:18 PM EDT

As long as Virtualization can segment cores then I don't see the big issue right now. Yes, the wall was hit with the integrated manufacturing processes of today but that doesn't mean the wall will be their tomorrow. On top of that, many a hardware geek has super-cooled current processors and gotten an almost 200% speed increase. Maybe people are just overlooking trying to scale better cooling solutions.

Sign in to Reply

aivchenko

5/4/2011 5:10 PM EDT

The problem is not so much in programming for parallel processes (it can be successfully done in Verilog) but rather that the single-thread Turing model is not very good for parallelism.
And even bigger problem to overcome is the human brain which can manipulate with 5 to 7 objects at a time.
So without development of the new tools it is highly improbable that multicore approach will allow continuations of the Moore law

Sign in to Reply

Rishiyur.Nikhil

5/5/2011 2:12 PM EDT

Re. "A tool should be able to find and point out the dependency that keeps tasks A and B from running in parallel." This problem is almost as old as computing itself. In the 1960s, when "vector machines" first appeared (CDC 6600, Cray 1), this was called the "dusty deck" problem, i.e., couldn't some tool take our existing Fortran codes ("dusty decks") and automatically parallelize them to run on the new vector machines? 50 years of research return an emphatic "No!". The first issue is that parallel algorithms are often different from sequential algorithms; you've already lost the game if you start with sequential algorithms. Second, the dependency analysis is simply computationally intractable, except in so-called "loop-and-array" codes: well-structured, properly nested FOR-loops with affine array indexes--this is a very niche success.

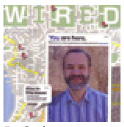
Sign in to Reply

Rishiyur.Nikhil

5/5/2011 3:45 PM EDT

Re. 'we already have too many of them [programming languages]. "First need to understand what it means to be parallel".': Perhaps these are not separate concerns. George Boole said, in his "Investigation of the Laws of Thought", that, "language is an instrument of human reason, and not merely a medium for the expression of thought". See also Ken Iverson's 1979 Turing Award lecture, "Notation as a Tool of Thought". Our "too many" sequential languages fundamentally restrict how we even think about computation.

Sign in to Reply


DrQuine

5/6/2011 7:32 PM EDT

Certainly there are processes running on computers (graphic display, background virus checking, decryption, document scanning, PDF creation, music playing) that could be allocated to separate cores and then free up the main thread to run faster. When I look at my sluggish computer, many of the dozens of threads that are running could be off-loaded. Even if everything cannot be perfectly parallel processing optimized, every bit helps. Also, when systems get into a gridlocked state, it would seem an ideal opportunity to instantly spawn a parallel process and return some control to the user.

Sign in to Reply

Ian Joyner

5/13/2011 9:36 PM EDT

Mapou, I believe we are talking more about the von Neumann model rather than Turing, who was much more a visionary in complex interactions in systems, although he did introduce the very simple Turing machine to reason about computability.

John Backus asked the question "Can we be liberated from the Von Neumann Style":

http://www.stanford.edu/class/cs242/readings/backus.pdf

He spent the rest of his life searching for this.

O do kind of agree with you sentiments though. We have had a generation of computer scientists who ignored what went on in Burroughs machines while they chased low-level performance via RISC. We should become reacquainted with this architecture and its multiprocessor/multiprocessing capabilities:

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

Bob Barton who designed these machines would also share your sentiments since he wrote about the low-level cults that had become entrenched (in the 60s) rather than true system-level architecture. There are a few papers of his around the web.

Sign in to Reply

Please sign in to post comment

Submit Comment Follow Comments