Observations The world is yet to see a truly successful general-purpose parallel computer for single task completion time

~2003 Wall Street traded companies desperate enough to give up the safety of the only paradigm that worked for them for parallel computing.

2008 Not (yet?) desperate enough to follow knowledge and understanding (theory and practice) of ~40 years of parallel computing. Mystified by urge to repeat the same mistakes expecting different results ☺ ☺

If this presentation will not offend you, please talk to me after the panel ☺ ☺ ☺ not trained to explain/treat this urge condition

Proposed QUESTION: can such a computer be developed?

Uzi Vishkin

UMIACS University of Maryland Institute for Advanced Computer Studies

A. JAMES CLARK SCHOOL of ENGINEERING

UNIVERSITY OF MARYLAND 1856

- History
  repeats itself first as
- Tragedy
  second as
- Farce
  (improbable plot)
  --Karl Marx

…we manage to make even Marx look champion of effectiveness

Thread 1 [Led to: To many users programming existing parallel computers is "as intimidating and time consuming as programming in assembly language", NSF Blue-Ribbon Panel on Cyberinfrastructure]

Build-first, figure-out how to program later (- 1990)

Failed: rather limited use, in spite of huge effort; … some programming language emerged (~1990s)

Improbable

1. Use proven (general-purpose) parallel architecture failures as basis for building multi-cores
2. Rush to standardize languages before having at least a successful architecture prototype
3. Ignore algorithms technology. Algorithms: Only CS field where winner is undisputed. What will your multi-core approach teach in the algorithms course?

Serial algorithms: big success. PRAM algorithms: (i) natural extension of serial algorithms; (ii) won every idea battle; others lost even when PRAM was absent! (iii) finally can be built (UMD PRAM-On-Chip)

Impasse

All vendors committed to multi-cores. Yet, their architecture and how to program them for single task completion time not clear (are they clueless?) ➔Avoid investment in long-term application SW development since: (i) architecture is about to change, and/or (ii) may bet on the wrong horse

You may want to ask why core CS algorithms people are not participating today? Answer: they have little new to add.

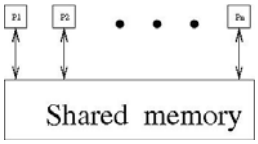
Would you believe that 19 of 38 participants of IBM-NSF 12/1988 workshop on directions for parallel computing were algorithms/theory people?!

So, what was the main message of these people back then?

Thread 2 History Let's first figure out how to think algorithmically in parallel. Build later.

Parallel Random-Access Machine/Model (PRAM)

Abstraction Concurrent accesses to memory, same time as one

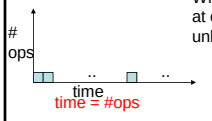


Lots of detail (e.g., how to allocate each among p processor to a task)

Still, architects considered the PRAM way too simple

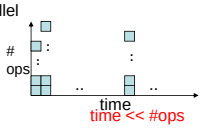
Conjecture [SV82] It is enough to do this:

Serial doctrine



What could I do in parallel at each step assuming unlimited hardware ➔

Natural (parallel) algorithm



Just be sure that $\#ops_{parallel} \sim \#ops_{serial}$

The rest (a full PRAM algorithm) is just a matter of skill.

Postscript: lots of evidence that this "work-depth methodology" works. Used as framework in PRAM algorithms textbooks: JaJa-92, Keller-Kessler-Traeff-01

The PRAM Rollercoaster ride

Late 1970's Dream

HISTORY Won the battle of ideas on parallel algorithmic thinking. No silver or bronze! Model of choice in all theory/algorithms communities. 1988-90: Big chapters in standard algorithms textbooks. "Nothing could stop it…"

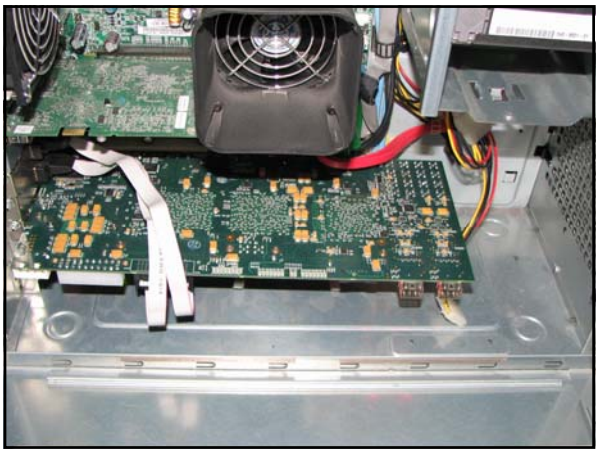
TRAGEDY FCRC'93: "PRAM is not feasible". BUT, even the 1993+ despair did not produce proper alternative➔ Not much choice beyond PRAM!

Dream coming true? PRAM-On-Chip vision: 64-processor eXplicit-multi-threaded (XMT) FPGA-prototype (not simulator) @UMD. SPAA'07, CompFrontiers'08.

How come: crash course on parallel computing

– How much processors-to-memories bandwidth?

Enough	Limited
Ideal Programming Model: PRAM	Programming difficulties

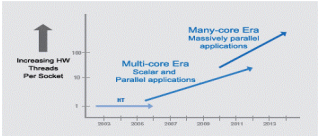


5-year old impasse

Chapter 1 1946➔2003: Serial. 5KHz➔4GHz.

Chapter 2 2004--: Parallel. #"cores": ~d^{y-2003}

Source: Intel Platform 2015
Date: March 2005



Since 2003 clock frequency growth is flat. HP-07:

If you want your program to run significantly faster … you're going to have to parallelize it ➔ Parallelism: only game in town

#Transistors/chip 1980➔2011: 29K➔30B!

When (NOT if) can we start teaching parallel algorithms & programming at all level? Brace for malpractice suits if not:

22-year old dinosaurs trained for 50-year career dominated by parallelism by programming yesterday's computers. We don't only under-teach: we mis-teach bad serial habit

"Informed" conjecture: must teach in CS Freshman year

Impasse excuse

All vendors committed to multi-cores. Yet, their architecture and how to program them for single task completion time not clear ➔ But, how do we know which approach to teach?! Like SW vendors, our preferred course of action is to wait and see who emerges as winners.

Nice try, but:

1. What about graduates before this is resolved?
2. When resolved: will you not teach the basics? The PRAM level of cognition is necessary, as it falls in the common denominator of other approaches. (It is also sufficient for a real architecture…)
3. Demonstrated: PRAM-like programming can be taught to high-school students (F'07) & non-CS majors (S'08): "CS&E is where the action is!"; compare appeal with malpractice suit

Conclusion

Any successful general-purpose approach must (also) answer: what will be taught in the algorithms class? Otherwise dead-end

PRAM: only current answer

PRAM-On-Chip: Showing how PRAM can pull it

Culler-Singh 1999: "Breakthrough can come from architecture if we can somehow…truly design a machine that can look to the programmer like a PRAM"

Final thought

Meritocracy will lead us to a solution and away from a farce. Not an …ism dogma or business cartels. This is up to **us:**

IPDPS09? *Organize (fair) competition among solutions*

Q&A

Question: Where can I find more information

Answer: www.umiacs.umd.edu/users/vishkin/XMT/index.shtml

Question: Why PRAM-type parallel algorithms matter, when we can get by with existing serial algorithms, and parallel programming methods like OpenMP on top of it?

Answer: With the latter you need a strong-willed Comp. Sci. PhD in order to come up with an efficient parallel program at the end. With the former (study of parallel algorithmic thinking and PRAM algorithms) high school kids can write efficient (more efficient if fine-grained & irregular!) parallel programs.

Some Quotes

- *The single-chip supercomputer prototype built by Prof. Uzi Vishkin's group uses rich algorithmic theory to address the practical problem of building an easy-to-program multicore computer. Vishkin's XMT chip reincarnates the PRAM algorithmic technology developed over the past 25 years, uniting the theory of yesterday with the reality of today.* Charles Leiserson, MIT, 2007.
- *I am happy to hear of the impending announcement of a 64 processor prototype of Uzi Vishkin's XMT architecture. This system represents a significant improvement in generality and flexibility for parallel computer systems because of its unique ability to exploit fine-grain concurrency. It will be able to exploit a wider spectrum of parallel algorithms than today's microprocessors can, and this in turn will help bring general purpose parallel computing closer to reality.* Burton Smith, Microsoft Technical Fellow, 2007.
- *Today's multi-core processors support coarse grain parallelism. Professor Vishkin has defined a new parallel architecture that supports extremely fine-grained threading. On XMT, a program can be profitably decomposed into tasks as small as 10 instructions. With a complete programming model and an impressive FPGA-based prototype, Professor Vishkin is proposing a compelling alternative design for future microprocessors.* Geoff Lowney, Intel Fellow, April 2008.