IPDPS Looking Back Panel

Uzi Vishkin, University of Maryland

UMIACS University of Maryland Institute for Advanced Computer Studies

A. JAMES CLARK SCHOOL *of* ENGINEERING

UNIVERSITY OF MARYLAND 18 56

From 30K feet

- *What has gone well? What has gone wrong?*
  Certain esoteric* developments
- *What has gone really wrong?*
  Lack of exoteric** validation

* Requiring/exhibiting knowledge that is restricted to a small group

** Suitable to be imparted beyond this small group

➔ **I would like to see (for example) 9th graders in HS**, not IPDPS authors, **rank multi-cores for achieving hard speedups**

➔ **Ease-of-programming & speedups will guide them**

- *What came as a surprise?*
  The low level of attention to exoteric validation by all:
  - Academia,
  - Government, and most surprising
  - Industry

2

Joint UIUC/UMD Parallel Algorithms/Programming Course

David Padua, University of Illinois at Urbana-Champaign
Uzi Vishkin, University of Maryland
Jeffrey C. Carver, University of Alabama

UMIACS University of Maryland Institute for Advanced Computer Studies

A. JAMES CLARK SCHOOL *of* ENGINEERING

UNIVERSITY OF MARYLAND 18 56

Assumption If **you** want your program to run significantly **faster** … **you're going to have to parallelize it**, HP, 4th edition, 2007

- you're going to have to parallelize it .. But what does it mean?
  * Identify concurrent operations, or
  * Program for locality (decomposition-first)
- faster in what way?
  * Asymptotic speeds WRT to serialized version, or
  * Hard speedups WRT best serial
- who is you?
  * Exoteric versus esoteric: 9th graders in HS, or IPDPS authors

My proposed solution

1. Rank multi-cores for achieving hard speedups

2. Get out of the esoteric bubble. Have the ranking done by the **broadest** (most exoteric) circle possible

What has gone wrong 1

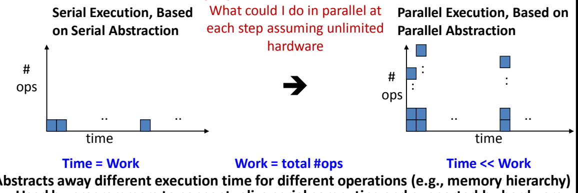
The Trouble with Multicore: Chipmakers are busy designing microprocessors that **most programmers can't handle** —*D. Patterson, IEEE Spectrum 7/2010*

Only heroic programmers can exploit the vast parallelism in current machines
– *The Future of Computing Performance: Game Over or Next Level?, Report by CSTB, NAE 2011*

If the objective is bringing parallel computing to the mainstream CS:
- **Too difficult to achieve speedups** on all parallel machines to date
- **Failed** to generate a broad base of **application programmers**

Serial Abstraction & A Parallel Counterpart

- **Rudimentary abstraction that made serial computing simple:** ***that any single instruction available for execution in a serial program executes immediately – "Immediate Serial Execution (ISE)"***


Serial Execution, Based on Serial Abstraction
What could I do in parallel at each step assuming unlimited hardware
Parallel Execution, Based on Parallel Abstraction
# ops
time
# ops
time
Time = Work
Work = total #ops
Time << Work

Abstracts away different execution time for different operations (e.g., memory hierarchy) Used by programmers to conceptualize serial computing and supported by hardware and compilers.

- **Rudimentary abstraction for making parallel computing simple:** ***that indefinitely many instructions, which are available for concurrent execution, execute immediately,*** **dubbed Immediate Concurrent Execution (ICE) # processors not even mentioned. Falls back on the serial abstraction if 1 instruction/step.**

V: Using Simple Abstraction to Reinvent Computing for Parallelism, CACM, Jan 2011

What has gone well

- Parallel PRAM algorithmic theory, second in magnitude only to the serial algorithmic theory
- Won the "battle of ideas" in the 1980s. Repeatedly challenged without success, since then ➔

Robust! Must take it into account in architecture specs .. but only if want the architecture to succeed

7

Feasible for many-cores

Algorithms

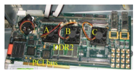
Programming

Programmer's workflow

Rudimentary yet stable **compiler**

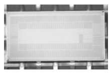
PRAM-On-Chip HW Prototypes

64-core, 75MHz FPGA of XMT [SPAA98..CF08]



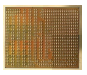
Toolchain Compiler + simulator HIPS'11

128-core interconnection network



IBM 90nm: 9mmX5mm, 400 MHz [HotI07]

FPGA design➔ASIC



IBM 90nm: 10mmX10mm

150 MHz

Architecture scales to 1000+ cores on-chip

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

What has gone wrong 2: Why most programmers can't handle today's machines?

1. Mismatch of architectures to algorithms

2. Flawed architecture foundation

- originated with 'design-first figure-out-how-to-program-later'

- Where are the *rewards*?

1. Funding for new general-purpose architectures: basically gone
2. Originality-seeking publications culture ➔ mismatch provides rich opportunities; flawed system legitimate if vendor-backed
3. Easy-to-achieve, strong speedups are almost non-publishable

9

What came as a surprise

A fool may throw a stone into a well which a hundred wise men cannot pull out

But:

☺ The wise men can write many papers on efforts

☺ Caveat need fresh supply of stones/wells for intellectual merit

☺ Our brilliant solution yet another machine too difficult to program

The surprise

We exceeded the imagination of the greatest philosophers of science on eccentricity of scientific communities … making Ludwik Fleck blush

Not easy Fleck coined the term 'thought collectives' in 1935, 27 years before Thomas Kuhn's Structure of Scientific Revolutions

10

What is the solution?

- In the science enterprise: "relatively small esoteric circles of experts and much bigger exoteric circles of school teachers and people applying scientific knowledge in practice"
- "The thought collective can work efficiently **only** when it gets suitable encouragement or stimuli from the exoteric circles of science"

➔Enough hiding. Got to broadly rank systems by 'achieving speedups'

11

Example for evidence on **ease of obtaining speedups**

Breadth-first-search (BFS)

- 40+ students in fall 2010 joint UIUC/UMD course
- <1X speedups using OpenMP on 8-processor SMP
- 8x-25x speedups on 64-processor XMT FPGA prototype.

But, what's the big deal of 64 processors beating 8?

- Silicon area of 64 XMT processors ~= 1-2 SMP processors

12