Lesson 12

# Current Research Course Review

*[Pat 08] [A-TKS 07] [KCHK 06]*

Moore’s Law

Multiprocessor Challenge in 1980’s

Multicore Challenge Now

Course Review

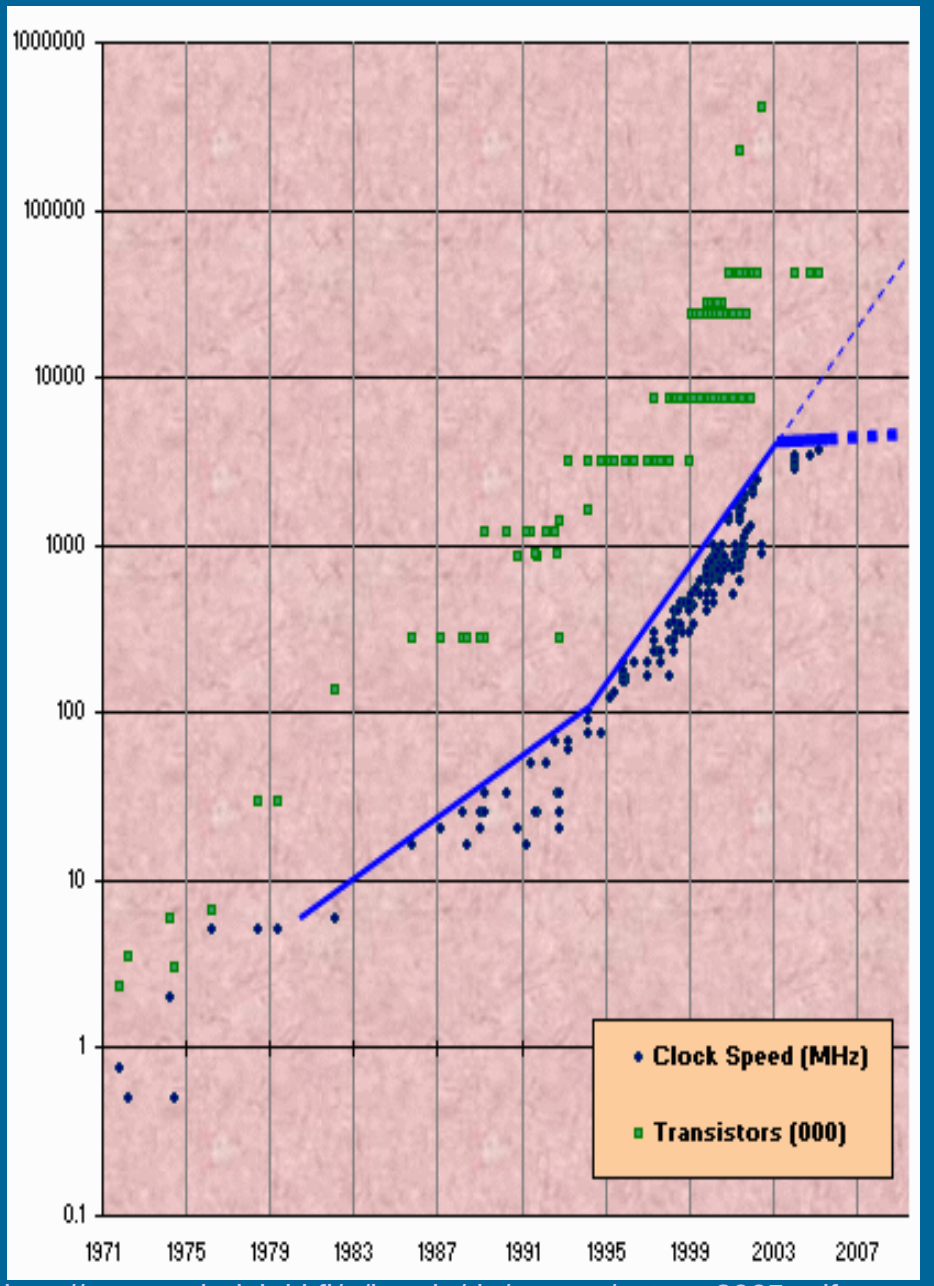
# Moore's Law

- *"The number of transistors that can be inexpensively placed on an integrated circuit is increasing exponentially, doubling approximately every two years"* (orig. 18 months)
  - Gordon E. Moore, 1965
  - Memory size in chips will double every two years
  - Increase in transistor count is also a rough measure of computer processing performance, resulting to processing speed doubling every two years
- "Heat barrier" limit for processing speed reached 2004
  - Luke Collins, IEE Review, Jan 2003

http://ieeexplore.ieee.org/iel5/2188/26844/01193720.pdf

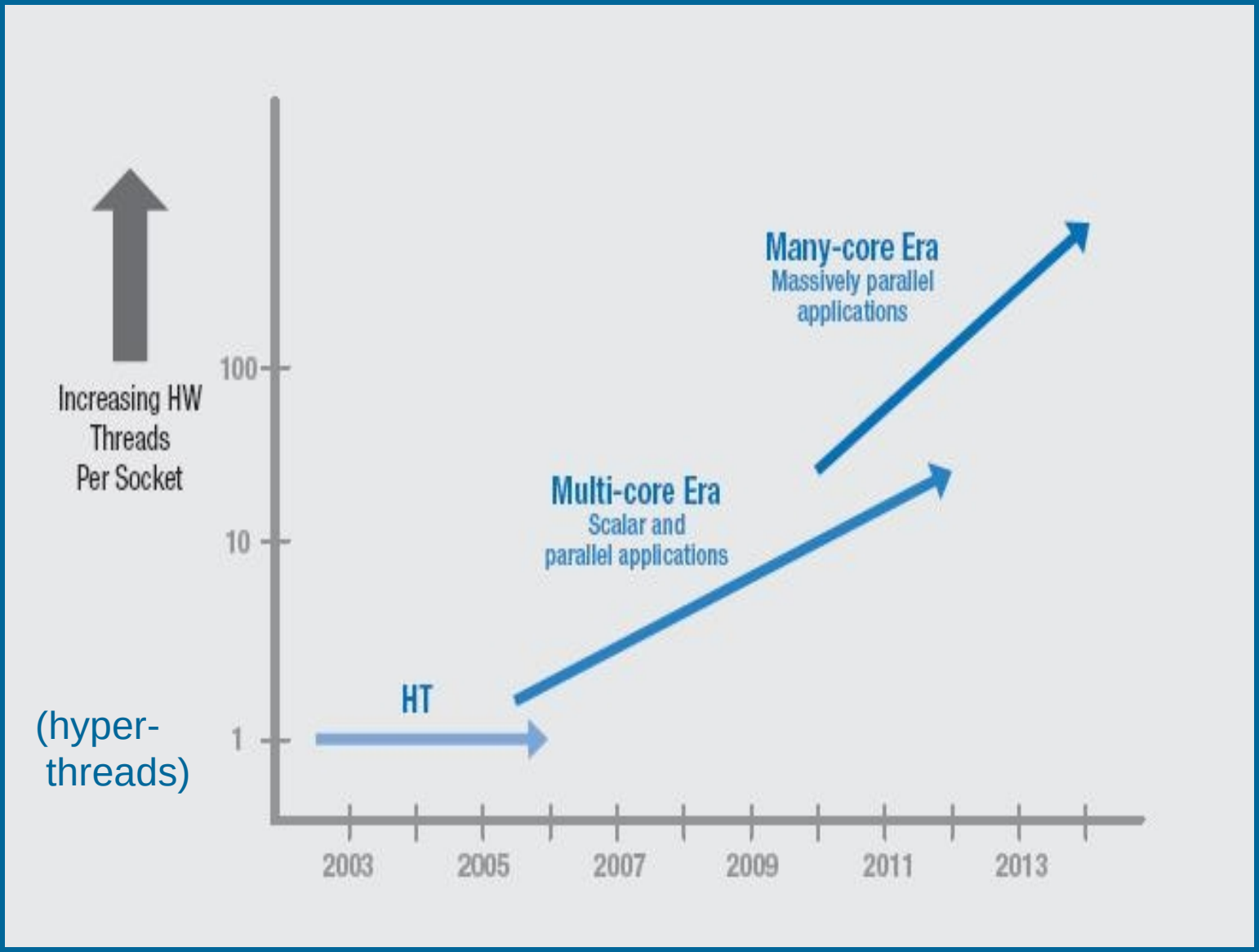
# Problem

- Moore's Law will not give us faster processors (any more)
  - But it gives us now more processors on one chip
    - Multicore CPU
    - Chip-level multiprocessor (CMP)




Herb Sutter, "A Fundamental Turn Toward Concurrency in SW", Dr. Dobb's Journal, 2005.

http://www.cs.helsinki.fi/u/kerola/rio/papers/sutter_2005.pdf




(hyper-
threads)

Borkar, Dubey, Kahn, et al. “Platform 2015.” Intel White Paper, 2005.

http://www.cs.helsinki.fi/u/kerola/rio/papers/borkar_2015.pdf

# Moore's Law Reinterpreted

- Number of cores per chip doubles every two years, while clock speed decreases
  - Need to utilize systems with hundreds or thousands of cores
  - Need to handle systems with millions (billions?) of concurrent threads
  - Need to emphasize scalability – not best performance for fixed number of cores.
  - Need to be able to easily replace inter-chip parallelism with intra-chip parallelism

Marc Snir

http://www.cs.helsinki.fi/u/kerola/rio/papers/snir_2008.pdf

# Multi-core: An Inflection Point also in SW Development

- Multi-core architectures: an inflection point in mainstream SW development
- Writing parallel SW is hard
  - Mainstream developers (currently) not used to thinking in parallel
  - Mainstream languages (currently) force the use of low-level concurrency features
  - Must have with new systems?
- Navigating through this inflection point requires better concurrency abstractions

# The Multicore Challenge

- Heat barrier dead-end for chip speed
- So, try now multicore chips …
  - Shared memory multiprocessor
- Similar multiprocessor HW tried before (not at chip level)
  - Convex, Floating Point Systems, INMOS, Thinking Machines, nCUBE, Kendall Square Research, MasPar, Encore, Sequent, …
  - All failed – Patterson's *"Dead Parallel Computer Society"*
- John Hennessy:
  - *"…when we start talking about parallelism and ease of use of truly parallel computers, we're talking about a problem that's as hard as any that computer science has faced. … I would be panicked if I were in industry."*

http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=445&page=4

- Challenge: How to use multicore effectively
- Answer: Industry/government funded research?
  - Scale: Manhattan Project

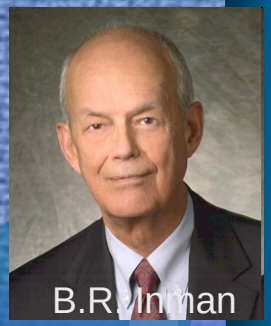
http://www.cccblog.org/2008/08/26/the-multicore-challenge/

# MCC 1983 - (2000)

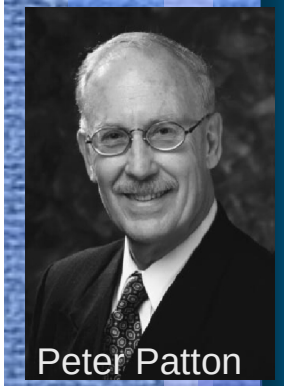
- re: Japanese 5th Generation Project
  - Ministry of Internat. Trade and Industry (MITI)
  - Japan, 1982, 10 years, $850M
  - New type of computer to run Artificial Intelligence applications
- US national level response to Japanese MITI Project
  - Massively Parallel Processing (MPP), how to build and use them
- Microelectronics and Computer Technology Corporation (MCC), Austin, Texas
  - Some 450 researchers in 1986
  - 12 companies in 1983: Control Data, DEC, Harris, RCA, Sperry-Univac, NCR, Honeywell, National Semiconductor, Advanced Micro Devices, Motorola, …
  - More companies later on: Microsoft, Boeing, GE, Lockheed, Martin Marietta, Westinghouse, 3M, Rockwell, Kodak, Nokia 1997, DoD, …
  - Budget $50-100M per year, for 20 years?

http://www.tshaonline.org/handbook/online/articles/MM/dnm1.html

# MCC 1983 - (2000)


B.R. Inman

- CEO Admiral Bobby Ray Inman (ex NSA #1, ex CIA #2)
  - Well connected, new law to avoid antitrust problems 1984
- Main Research Areas (Programs)
  - Software technology, semiconductor packaging, VLSI computer-aided design, parallel processing, database management, human interfaces and artificial intelligence/knowledge-based systems
- Parallel Processing Program
  - "*What type of MPP computer to build and how to use it?*"
  - VP Peter Patton 1983-86, VP Stephen Lundstrom 1986-87
  - Some 30 researchers for 3 years, lots of resources
  - Lots of published (and proprietary) research papers
  - Reviews every 3 months, for knowledge transformation from MCC to shareholding companies
  - Could not formulate goal properly, starts to disintegrate 1987


Peter Patton


S. Lundstrom

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

# MPP Challenge in early 1980's

- Shared memory multiprocessors
  - With cache coherence
  - Memory bus can handle some 20 processors
- More processors have new problems
  - Interconnect networks
    - All-to-all, butterfly, cube connected cycles, hyper-cube, …
  - Multi-level memory, varying memory access times
    - Local, shared, node, network, …
- Operating systems
  - Shared Memory OS, Distributed OS?
  - Compiler technology – ouch!
- Applications tailored for just one system?

# The Multicore Challenge Challenge

- The Multicore Challenge
  - How to use multicore/shared-memory-multiprocessor effectively?
  - Answer: Industry/government funded research for many years (?)
- The Challenge Challenge
  - Is the Multicore Challenge the right challenge to take?
    - It might fail (again)
  - What other challenges would better suit our need for ever faster computers?

- The Real Challenge (?):
  - How to get computational speed to double every two years or every 18 months for all computations ? (just like before…)
    - Currently there is no answer
    - What if there is no answer ever? Is that ok?
  - Get computational speed to double every two years for some computations? Is this enough?
    - This is doable… but is it enough? Which "some"?

# Needs for Multicore Challenge

- How to synchronize processes in multiprocessor systems?
- How to make it easy to obtain parallel performance
- Scalable solutions (to many and even more processors)
- Avoid deadlocks
- Data consistency with error situations (abort support)
- Current programming languages force low level solutions
- Amdahl's Law: Proportion of serial code will give upper limit on speedup
  - With 5% serial code, max speedup is 20 (even for 100 processors)

$$\text{Speedup} = \frac{1}{(1-P) + P/N}$$

where P = parallel code proportion
N = nr of processors

Serial code proportion (1-P) should be almost zero. How?

# US Multicore Challenge Projects

- Target
  - 1000-core (or more) processors
  - Parallel algorithms, development environments, and runtime systems that scale to 1000s of hardware threads
  - Need new OS & HW architectures
    - What exactly is needed? That is the problem!
  - *What type of MPP computer to build and how to use it?*
- Tools
  - FPGA-based simulators to test out work
    - Field-Programmable Gate Array
    - Reprogram HW to test new HW ideas
- Funding problem – another challenge?
  - Defence Advanced Research Projects Agency (DARPA) funding receding in 2000-2008
    - Would have been needed big-time

# US Projects 2008

- Stanford University
  - Pervasive Parallelism Lab
- University of California at Berkeley
  - Parallel Computing Lab
- University of Illinois at Urbana-Champaign,
  - The Universal Parallel Computing Research Center
- The Multicore Association
  - Many companies share results

# Stanford


John Hennessy

- John Hennessy (Jan 2007):
  - "When we start talking about parallelism and ease of use of truly parallel computers, we're talking about a problem that's as hard as any that computer science has faced."
  - "I would be panicked if I were in industry."

http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=445&page=3

# Stanford

William Dally

John Hennessy

- 2008, $6M for 3 years,
- 9 faculty + 30 grad students
- Nvidia, Sun Microsystems, Advanced Micro Devices, Hewlett-Packard, IBM, Intel
- William Dally (chairman, Stanford CS dept)
  - Stream computing, transactional memory
- Enable use of parallelism beyond traditional scientific computing
- New ideas for high-level concurrency abstractions
- New ideas for hardware support for new paradigms

http://ppl.stanford.edu/wiki/index.php/Pervasive_Parallelism_Laboratory

http://ppl.stanford.edu/wiki/images/9/93/PPL.pdf

# Stanford

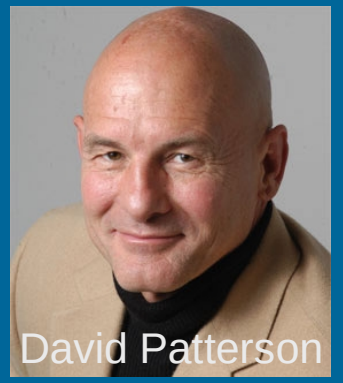

John Hennessy

- Goal: the Parallel Computing Platform for 2012
  - Make parallel programming practical for the masses — applications
  - Algorithms, programming models, runtime systems — programming and sw systems
  - architectures for scalable parallelism — architecture
    - 10,000s of HW threads
    - Parallel computing a core component of CS education
  - Build real, full system prototypes

# Universal Parallel Computer Research Centers (UPCRC's)

- Funding: Intel & Microsoft
  - Dan Reed (Microsoft, Extreme Comp Grp)
- Univ of California at Berkeley
  - Parallel Computing Lab
  - David A. Patterson
- Univ of Illinois at Urbana-Champaign
  - The Universal Parallel Computing Research Center
  - Marc Snir & Wen-mei Hwu

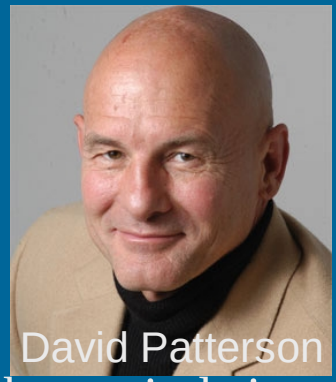

Dan Reed


David Patterson


Marc Snir


Wen-mei Hwu

# Berkeley


David Patterson

- David A. Patterson (Aug 2008):
  - "Knowing what we know today, if we could go back in time we would have launched a Manhattan Project to bring together the best minds in applications, software architecture, programming languages and compilers, libraries, testing and correctness, operating systems, hardware architecture, and chip design to tackle this parallel challenge."
  - "We need the US Government to return to its historic role to bring the many more minds on these important problem. To make real progress, we would need a long-term, multi-hundred million dollar per year program."

http://view.eecs.berkeley.edu/wiki/Main_Page

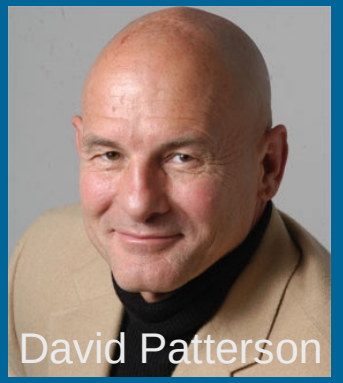
http://www.cccblog.org/2008/08/26/the-multicore-challenge/

# Berkeley

Parallel Computing Lab David A. Patterson

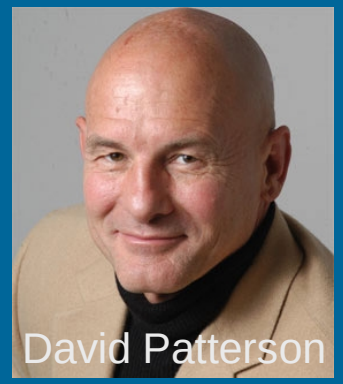
http://view.eecs.berkeley.edu/wiki/Main_Page

- David A. Patterson, 10 faculty + 40 grad students
- $17M  for 5 years
- Berkeley Emulation Engine v3 (BEE3)
- 7+ basic programming tasks at the heart of most parallel programs?
  - Health Care, Speech Recognition, New Music and Audio Technologies, Content-based Image Retrieval, Parallel Browser, Puppet-driven games, …
- Compositional verification and testing


David Patterson

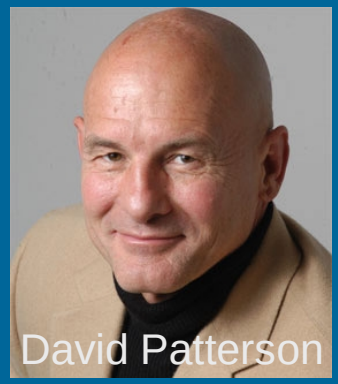
http://parlab.eecs.berkeley.edu/pubs/patterson-intro-20070604.ppt

# Berkeley


David Patterson

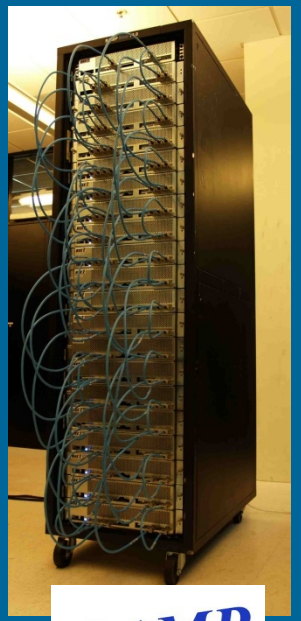
- Applications
  - Need new 21st century applications
    - Medicine, image, music, speech, …
- Computational bottleneck benchmarks
  - 7+ dwarfs, use them to analyse hw/sw designs
- Parallel SW development (55% faculty)
  - Implement 13 dwarfs as libraries or frameworks
  - Efficiency layer by experts (mutex, deadlock, etc)
  - Productivity layer by “normal” programmers
  - Create Composition and Coordination (C&C) language
  - 21st century code generation
- OS and Architecture
  - Very thin hypervisors

# Berkeley


David Patterson

- HW: build own academic Manycore
  - Research Accelerator for Multiple Processors
  - 4 FGPAs/board, 21 boards (84 FGPA'a)
- 1008 Core RAMP Blue
  - 12 32-bit RISC cores / FPGA
- Other architectures by FPGA redesign
  - RAMPants: 10 faculty
    - Berkeley, CMU, MIT, Stanford, Texas, Washington
    - Create HW&SW for Manycore community

Field Programmable Gate Array



# Illinois

## The Universal Parallel Computing Research Center (UPCRC)


- Marc Snir (Nov 2008):
  - "It is possible that parallel programming is inherently hard, in which case, indeed the sky is falling."
  - "An alternative view is that, intrinsically, parallel programming is not significantly harder than sequential programming; rather, it is hampered by the lack of adequate languages, tools and architectures."

http://www.cccblog.org/2008/11/17/multi-core-and-parallel-programming-is-the-sky-falling/

# Illinois

http://www.upcrc.illinois.edu/

- Marc Snir & Wen-mei Hwu
- $17M for 5 years
- Parallel programming can be (should be?) a child's play
- Simplicity is hard
  - Simpler languages + more complex architectures
  - a feast for compiler developers
- What hooks can HW provide to facilitate programming?
  - Sync primitives, debug/performance support

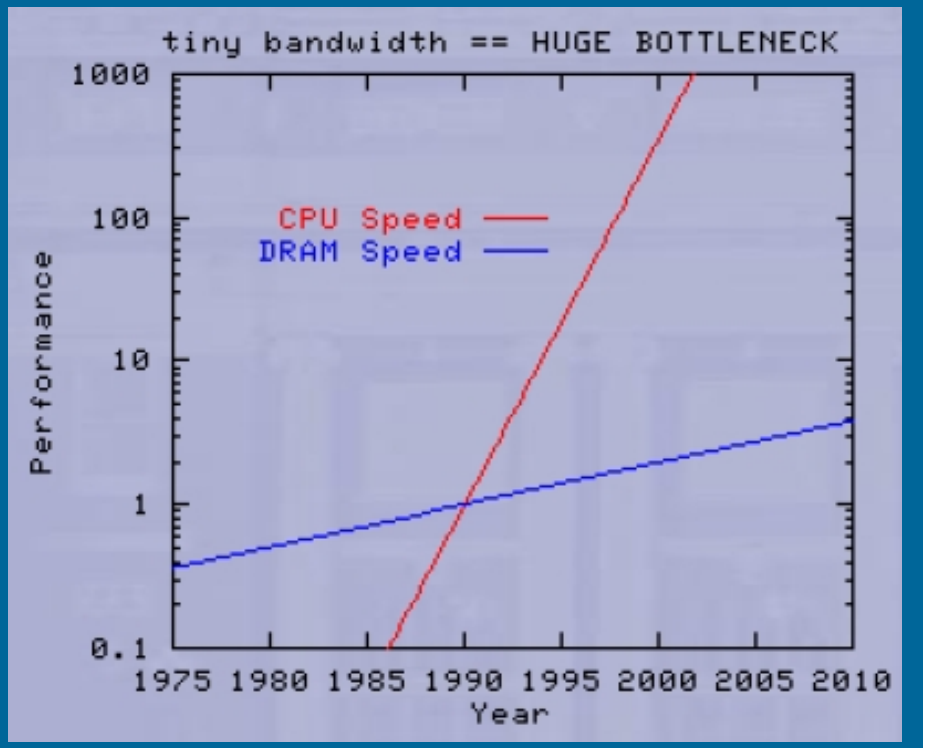

Marc Snir


Wen-mei Hwu

http://www.cs.helsinki.fi/u/kerola/rio/papers/snir_2008.pdf

# Illinois


- Moore's Law Reinterpreted
  - Number of cores per chip doubles every two years, while clock speed decreases
  - Need to be able to easily replace inter-chip parallelism with intra-chip parallelism
- Memory Wall
  - Most area and energy in chip budget is spent on storing and moving bits
- Reliability and Variance
  - MTTF (mean time to fail) per chip does not decrease – hardware is used to mask errors
  - Programmers do not have to handle faults



# Illinois

- New emphasis on deterministic (repeatable) parallel computation models – focus on producer-consumer or barrier synchronization, not on nondeterministic mutual exclusion
  - Simpler languages + more complex architectures
    = a feast for compiler developers
- Serial semantics, parallel performance model
  - Parallel algorithms are designed by programmers, not inferred by compilers

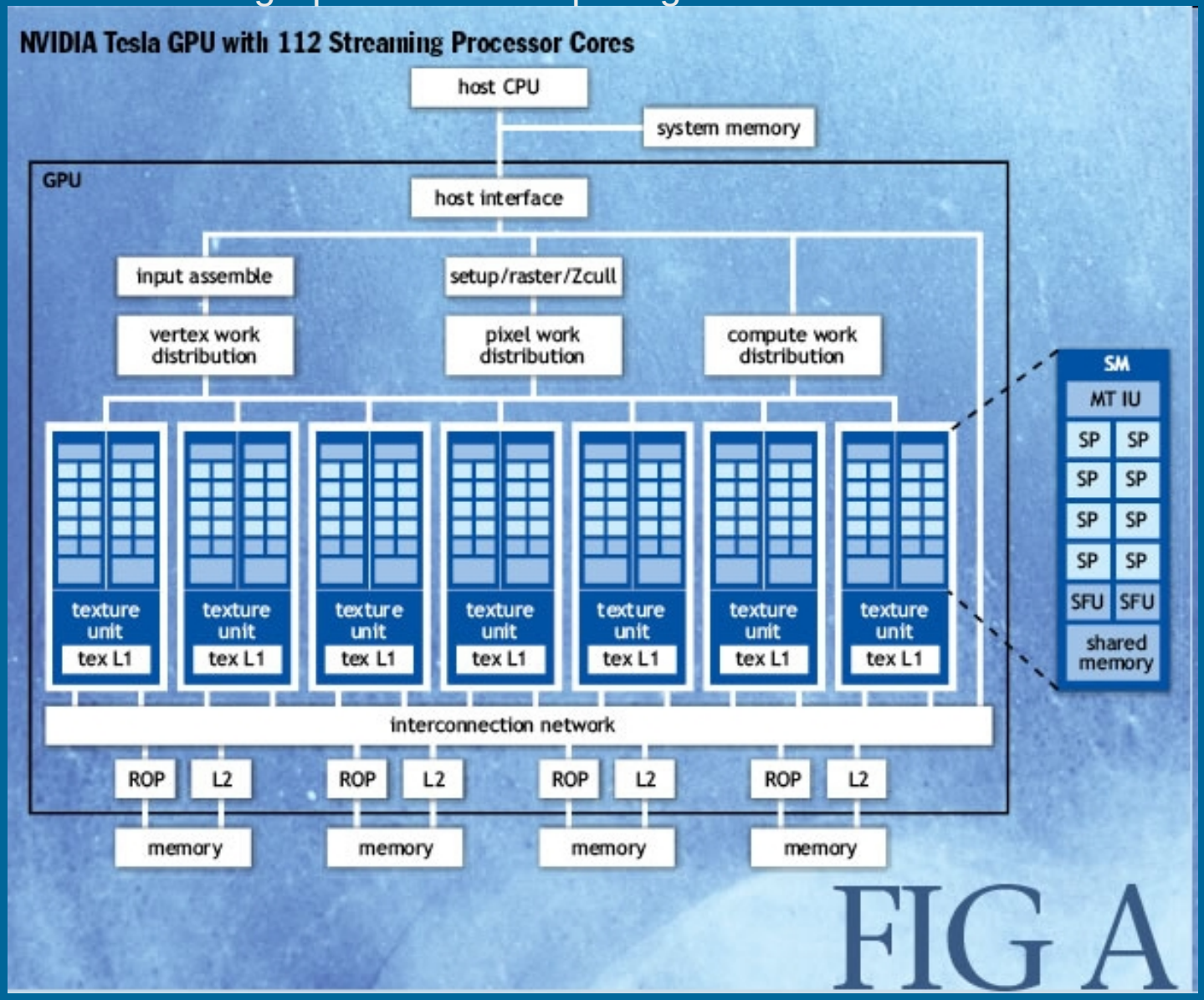

Wen-mei Hwu

- Every computer scientist educated to “think parallel”
  - Make parallel programming synonymous with programming
- What hooks can HW provide to facilitate programming?
  - Sync primitives, debug/performance support
- There is no silver bullet – no one technology solution

# The Multicore Association

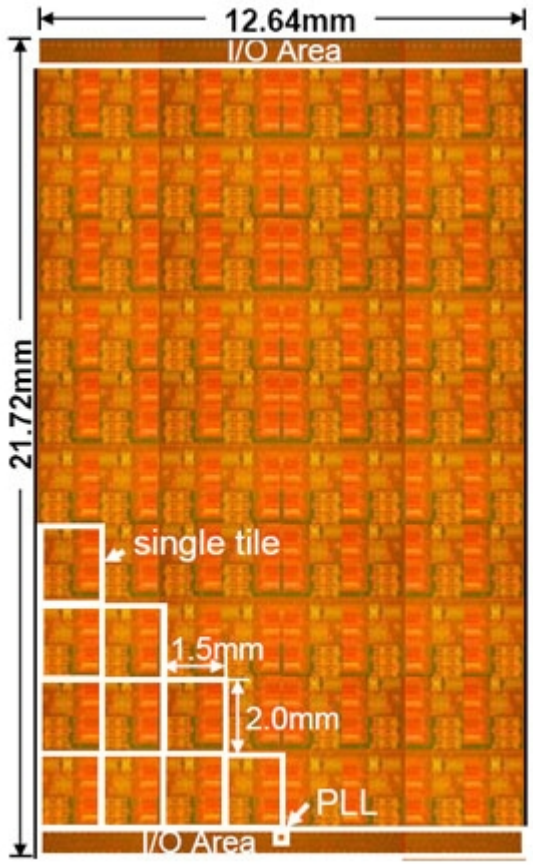
- Intel, NSN, Texas Instruments, Plurality, Wind River, PolyCore, Samsung, … http://www.multicore-association.org/home.php
- **Multicore Communications API** (MCAPI) work. gr (wg)
  - capture the basic elements of communication and synchronization for closely distributed embedded systems.
- **Multicore Programming Practices** (MPP) wg
  - develop a multicore software programming guide for the industry that will aid in improving consistency and understanding of multicore programming issues
- **Multicore Resource Management API** (MRAPI) wg
  - specify essential application-level resource management capabilities needed by multicore applications
- **Hypervisor** wg
  - support hypervisor (multiple OS's on same host) portability and multicore capabilities.

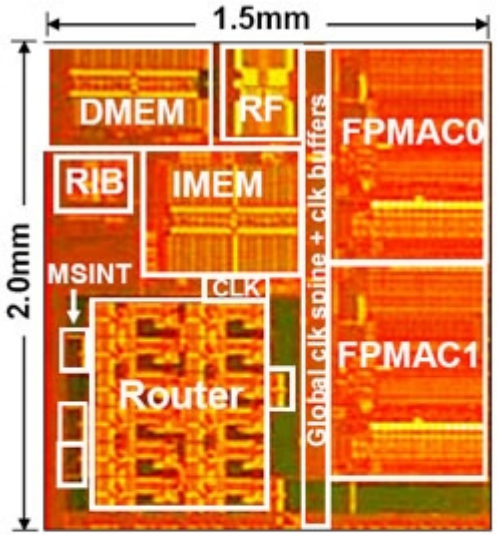
# Tesla unified graphics and computing architecture




http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=532&page=6

http://www.legitreviews.com/article/460/1/



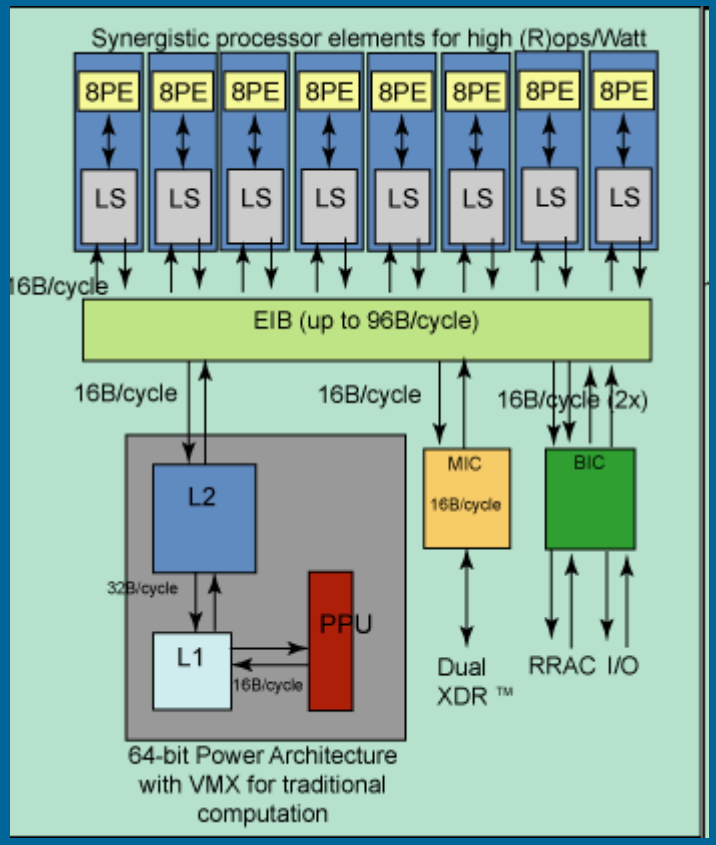

| | |
|---|---|
| Technology | 65nm CMOS Process |
| Interconnect | 1 poly, 8 metal (Cu) |
| Transistors | 100 Million |
| Die Area | 275mm² |
| Tile area | 3mm² |
| Package | 1248 pin LGA, 14 layers, 343 signal pins |

Intel Teraflops Research Chip wafer

Cool 80-core chip:

- block matrix operations
  1 TFLOPS at 1.0V at 110 °C

http://www.ibm.com/developerworks/power/library/pa-celltips1/




The Cell processor

Fast Roadrunner system

- 12 960 Cells, 1 PFLOPS, 2.3 MW
- 6 948 dual-core Opteron I/O
- total 116 640 cores
- 90 km fiber-optic cable, 500m²

# Likely Problems with Multicore Challenge

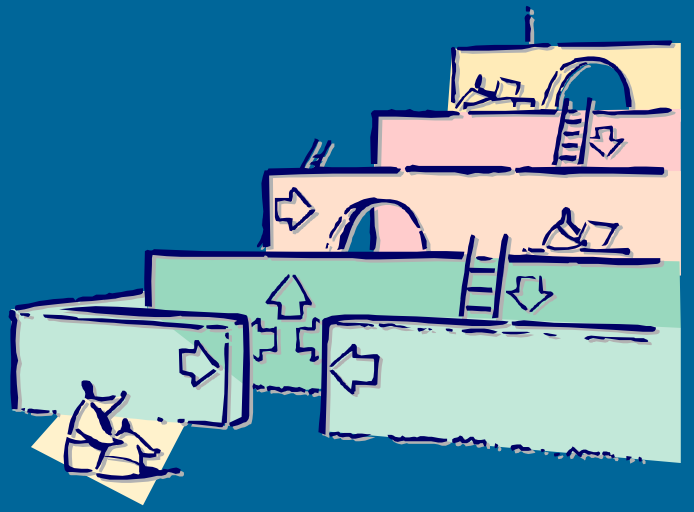
- Symmetric Multiprocessor (SMP) scales up to only somewhere (remember the 1980's)
- Must have interconnect networks
  - Between cores in chip, between chips, boards, and nodes
- How to distribute memory and access it
  - Local core memory in cache?
  - Memory hierarchy reworked? Disks disappear?
- How to implement I/O and database
  - How to distribute disks or other permanent stores
- Avoid communication & I/O bottlenecks
  - Remember Amdahl's Law
  - Minimize communication and serialization

# More Likely Problems with Multicore Challenge

- Computation + communication + memory use
  - Optimize on overall time/space?
- Test new processor architectures with FPGA's
- Operating system for new architectures?
- New languages for new architetures?
- Good compilers for new architectures?
- May end up with lots of different architectures
  - Masters & slaves, control hierarchies, ..
  - Applications run well only on one system?
    E.g., Systems based on STI cell vs. Intel 80-core chip?

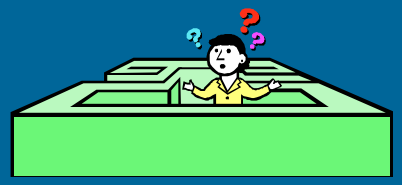
# Even More Likely Problems with Multicore Challenge

- Scalable processors architecture needs to be at least 3D?
  - Is 3D enough?
  - Stacked chips 2008
- Real stress to get results fast
  - Single-core dead-end?
  - New architectures designed and built now
  - Important applications built on new architectures



# Current Research Summary

- Moore's Law and what it means now
- What type of MPP computer to build and how to use it?
  - We have jumped into the multi-core train – is that OK?
- Some projects in Universities & Industry
  - Too small scale? $10M's but not $100M's or $1000M's …
- No silver bullet?
  - Get computational speed to double every two years for some computations? Is this enough?
- Make parallel programming synonymous with programming?
  - Only experts program the "efficiency layer" in SW?
- What happens if all these projects fail?
  - Get heterogeneous architectures that are incompatible with each other's code?



# Concurrent Programming with New Architectures

- Minimize synchronization and communication
  - Amdahl's Law
  - Barrier synchronization often good
  - Avoid any complex synchronizations that do not scale up
  - Mutual exclusion should not be used in computational work
- Use shared memory when possible
  - Faster than sharing data with messages
- How to partition problem so that overall solution time is minimized?
  - Distribute computing and data
  - Minimize communication and synchronization
  - Trade computing time to communication time
- Prepare for faults
  - Many components, some will fail, must be prepared

# Course Review

- Concurrency
  - Problems, atomic statements
  - Critical sections, synchronization, communication
  - What are the problems in writing concurrent programs?
- Disabling interrupts, busy wait, or suspension?
  - When to (not) use? HW vs. SW solution?
  - Shared memory or not? One or distributed system?
- Proofs of correctness
  - Functionality
  - Mutex, no deadlock, no starvation
  - How to do the proofs with temporal logic?
  - How to determine what you really are trying to prove?

# Course Review (contd)

- Deadlock
  - Detection, prevention, avoidance - DDA, Bankers
- Semaphores
  - Private semaphores, split semaphores, baton passing, implementation, busy-wait semaphores
- Monitors, protected objects
  - Condition variables, signal semantics
- Messages, RPC, channels, rendezvous
  - Concurrent algorithms
- Distributed mutex
  - (token passing) Ricart-Agrawala, Neilsen-Mizuno
- Current research

# Course Review (contd)

- Basic problems and solutions for them
  - Dining philosophers, sleeping barber, bakery
  - Readers-writers, producer-consumer
- Distributed system
  - Concurrency control mechanisms
  - Solutions for critical section problem

# What Should You Know?

- When to use what method for critical section (mutex), synchronization, or communication?
- How do you know you have a mutex problem?
- When would you use busy waits, semaphores, monitors, protected objects, RPC, channels, rendezvous?
- How do you implement XYZ with busy waits, semaphores, monitors, protected objects, RPC, channels, rendezvous?
- When is some technology not appropriate?

# What Should You Know?

- When do you need concurrent/distributed algorithms?
  - If serial solution is ok, use it!
- What type of OS/programming language library tools you have for CS/synchronization/communication problems?
- What do you need to study to solve your problem?
- What type of tools would you need to solve your problem?
- How does current research apply to me?
  - Do I need to study MPI (Message Passing Interface)?

# What Next at TKTL?

- How prove correctness of (concurrent) programs?
  - An Introduction to Specification and Verification

    Spesifioinnin ja verifioinnin perusteet
- Concurrency problems in distributed systems
  - Operating Systems

    Käyttöjärjestelmät
  - Distributed Systems

    Hajautetut järjestelmät
- Concurrency tools for Java programming
  - Software Design (Java)

    Ohjelmointitekniikka (Java)

# -- The End --

http://sti.cc.gatech.edu/SC07-BOF/06-Borrett.pdf

## Full Roadrunner Specifications:

6,912 dual-core Opterons
- 49.8 TF DP peak Opteron
- 27.6 TB Opteron memory

12,960 Cell eDP chips
- 1.33 PF DP peak Cell eDP
- 2.65 PF SP peak Cell eDP
- 51.8 TB Cell memory
- 277 TB/s Cell memory BW

3,456 nodes on 2-stage IB 4X DDR
- 13.8 TB/s aggregate BW (bi-dir) (1st stage)
- 6.9 TB/s aggregate BW (bi-dir) (2nd stage)
- 3.5 TB/s bi-section BW (bi-dir) (2nd stage)

432 10 GigE I/O links on 216 I/O nodes
- 432 GB/s aggregate I/O BW (uni-dir)
- (IB limited)