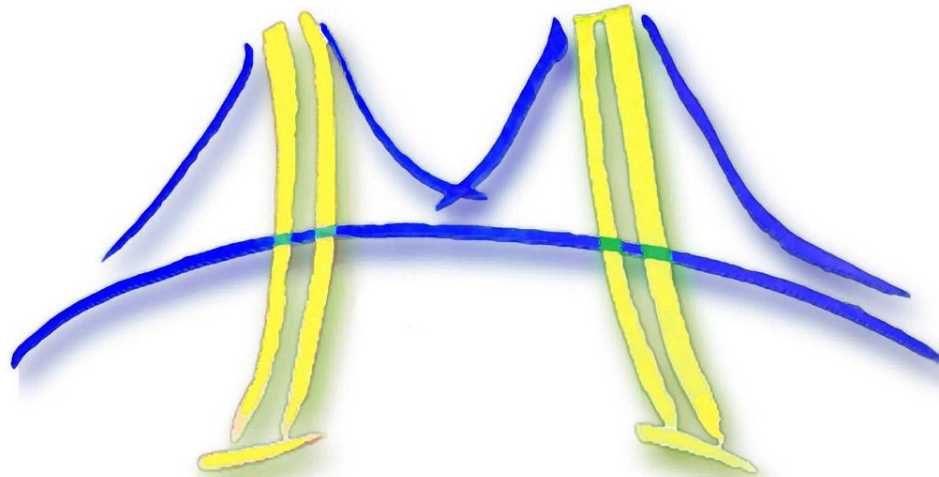


End of ParLab Celebration

May 30th, 2013



- ❖ 9:30 Where We Started, Where We Ended Up
- ❖ 11:15 Group Talks, Demos, and Testimonials
- ❖ 12:45 Lunch, Boxed Lunches
- ❖ 2:00 Group Talks, Demos, and Testimonials
- ❖ 4:30 Integrated Talk, Demo, and Feedback
- ❖ 5:50 David Wessel and Matthew Goodheart Concert
- ❖ 6:20 Group Photo
- ❖ 6:45 Reception, Dinner, and Toasts, Bancroft Hotel
- ❖ 10:00 Lubricated Discussion, Freehouse

- ❖ Sunglasses!
- ❖ No Food in Hertz Hall
 - Including coffee ☹️
- ❖ 15 Live Demos, 28 Speakers!
- ❖ Save Questions for the Breaks
- ❖ Videos to come

First Retreat – Jan 9-11 2008



January 2008

A black and white photograph of a clock face. The clock has a large outer dial and a smaller inner dial. The numbers 1 through 12 are visible on the outer dial. The text "JAN 2007" is overlaid in large, bold, white capital letters across the center of the clock face. The clock hands are visible, and the overall image has a vintage, slightly grainy appearance.

JAN 2007



The Berkeley View on the Parallel Computing Landscape

David Patterson, Krste Asanović, Kurt Keutzer,
and a cast of thousands
U.C. Berkeley

January 2007

- 5th Harry Potter movie opens July
- TV show *Mad Men* premieres July
- HP Laptop: 2 GHz Core 2 duo, 2 GB DRAM, 200 GB disk, \$1700
- *People* magazine Sexiest Man Alive: Matt Damon
- Maker best cell phone: Blackberry
- Maker best tablet: Microsoft



January 2007

High Level Message

- Everything is changing
- Old conventional wisdom is out
- We **desperately** need new approach to HW and SW based on parallelism since industry has bet its future that parallelism works
- Need to create a “watering hole” to bring everyone together to quickly find that solution
 - architects, language designers, application experts, numerical analysts, algorithm designers, programmers, ...

January 2007

Outline

- Old vs. New Conventional Wisdom
- 7 Questions to Frame Parallel Research
- New Benchmarks for New Architectures
- Hardware Building Blocks
- Human-centric Programming Model
- Innovating at HW/SW interface without Compilers
- Deconstructing Operating Systems
- Building innovative computers without custom chips
- Optimism for Parallel Computing Revolution?
- Where do we go from here?

January 2007

Conventional Wisdom (CW) in Computer Architecture

- 1. Old CW:* Power is free, but transistors expensive
 - *New CW* is the “Power wall”:
Power is expensive, but transistors are “free”
 - Can put more transistors on a chip than have the power to turn on
- 2. Old CW:* Multiplies slow, but loads and stores fast
 - *New CW* is the “Memory wall”:
Loads and stores are slow, but multiplies fast
 - 200 clocks to DRAM, but even FP multiplies only 4 clocks
- 3. Old CW:* We can reveal more ILP via compilers and architecture innovation
 - Branch prediction, OOO execution, speculation, VLIW, ...
 - *New CW* is the “ILP wall”:
Diminishing returns on finding more ILP

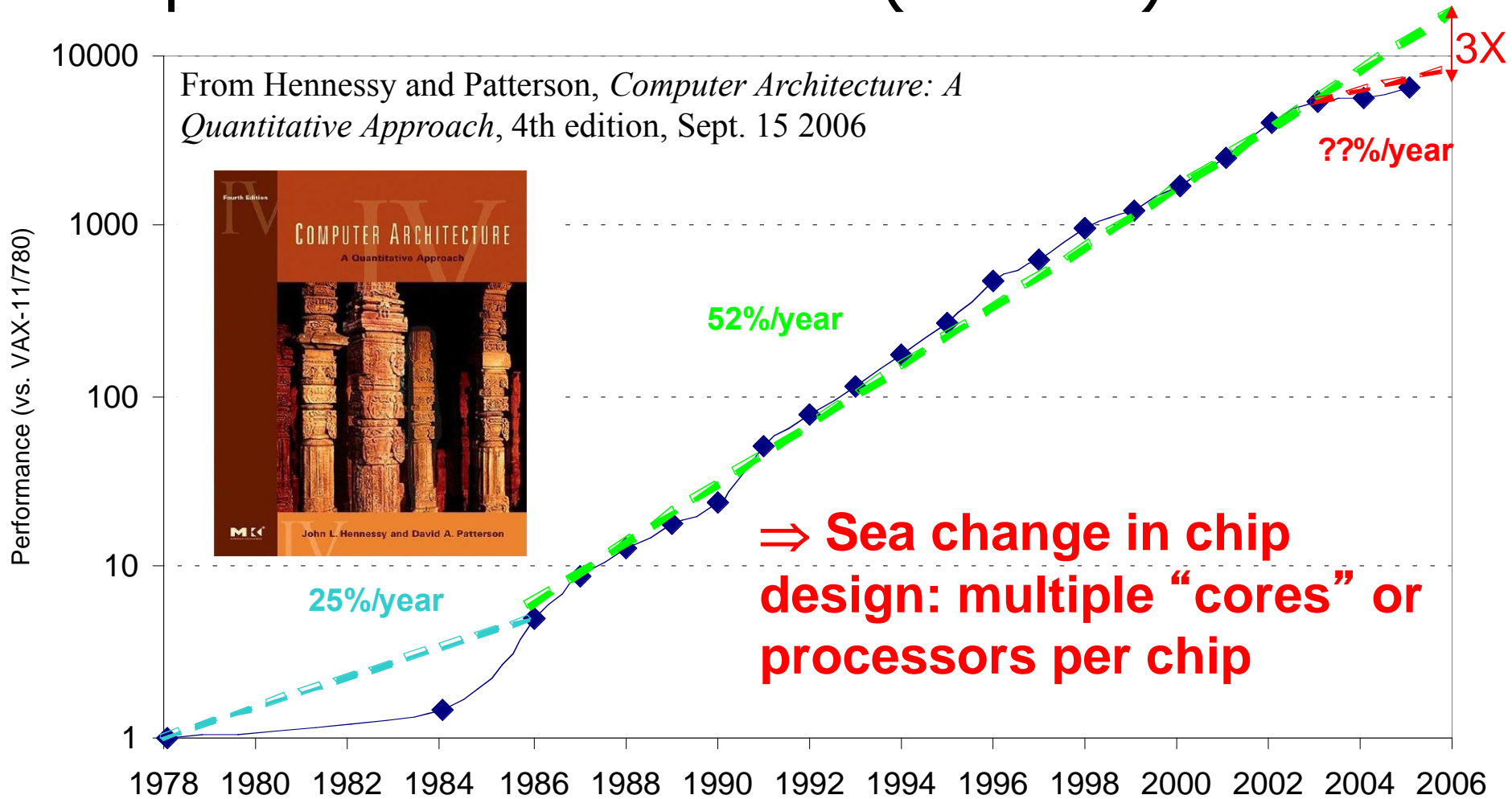
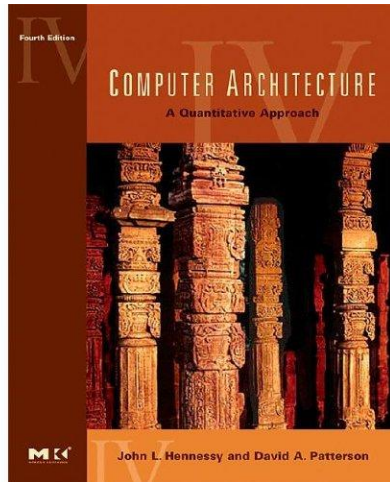
Conventional Wisdom (CW) in Computer Architecture

4. *Old CW*: 2X CPU Performance every 18 months
 - *New CW* is *Power Wall + Memory Wall + ILP Wall*
= *Brick Wall*
5. *Old CW*: Increasing clock frequency is primary method of performance improvement
 - *New CW*: Processors Parallelism is primary method of performance improvement
6. *Old CW*: Don't bother parallelizing app, just wait and run on much faster sequential computer
 - *New CW*: No one building 1 processor per chip
 - End of La-Z-Boy Programming Era

January 2007

Uniprocessor Performance (SPECint)

From Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 4th edition, Sept. 15 2006



- VAX : 25%/year 1978 to 1986
- RISC + x86: 52%/year 1986 to 2002
- RISC + x86: ??%/year 2002 to present

January 2007

Parallelism again? What's different this time?

“This shift toward increasing parallelism is not a triumphant stride forward based on breakthroughs in novel software and architectures for parallelism; instead, this **plunge into parallelism is actually a retreat from even greater challenges that thwart efficient silicon implementation of traditional uniprocessor architectures.**”

Berkeley View, December 2006

- HW/SW Industry bet its future that breakthroughs will appear in the not too distant future

January 2007

P.S. Parallel Revolution May Fail

- John Hennessy, President, Stanford University, 1/07:
*“...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.”*
“A Conversation with Hennessy & Patterson,” *ACM Queue Magazine*, 4:10, 1/07.

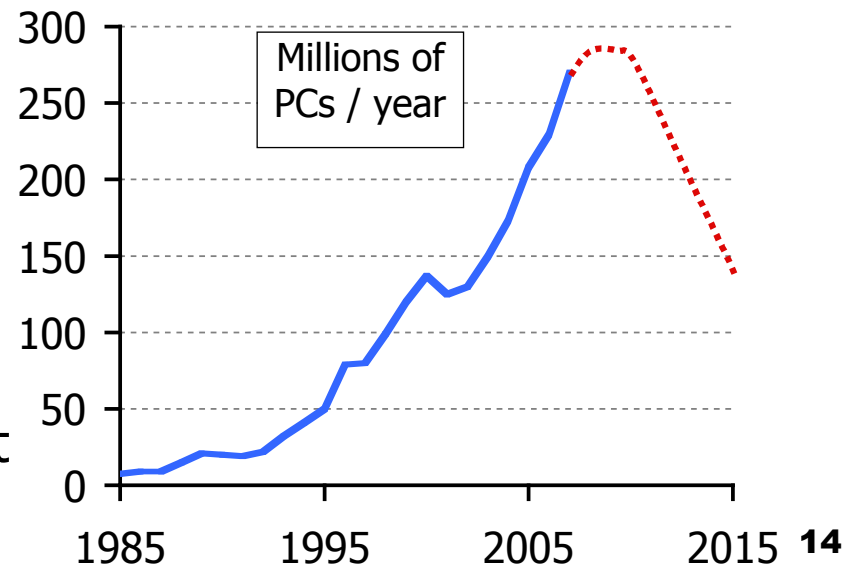
- 100% failure rate of Parallel Computer Companies

- Convex, Encore, Inmos (Transputer), MasPar, NCUBE, Kendall Square Research, Sequent, (Silicon Graphics), Thinking Machines, ...

- What if IT goes from a ***growth*** industry to a ***replacement*** industry?

- If SW can't effectively use 32, 64, ... cores per chip
⇒ SW no faster on new computer
⇒ Only buy if computer wears out

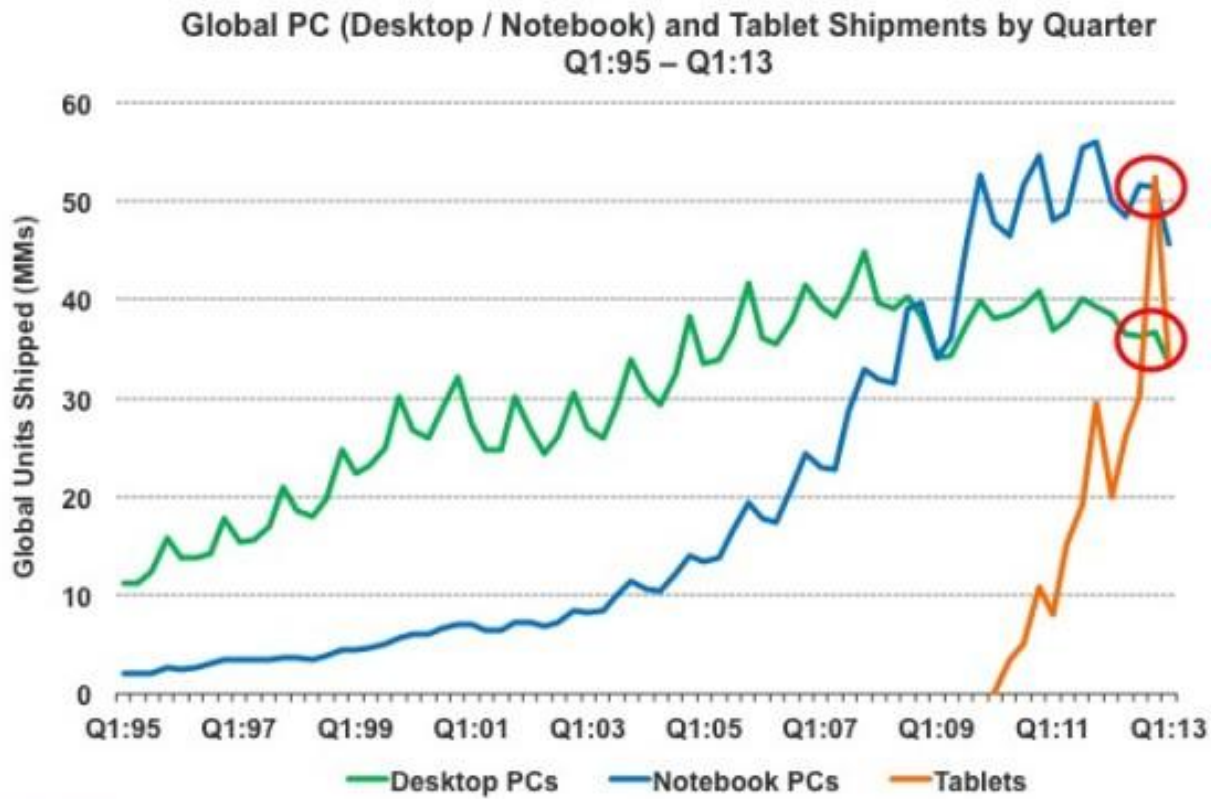
January 2007



PC quarterly sales plummet, sharpest drop on record

Tablet Shipments =

Surpassed Desktop PCs & Notebooks in Q4:12, < 3 Years from Intro

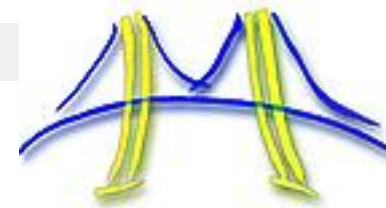


KPCB

Note: Notebook PCs include Netbooks.
Source: Katy Huberty, Ehud Gelblum, Morgan Stanley Research, Gartner, Data as of 4/13.

45

Need a New Approach



- Berkeley researchers from many backgrounds met between February 2005 and December 2006 to discuss parallelism
 - Circuit design, computer architecture, massively parallel computing, computer-aided design, embedded hardware and software, programming languages, compilers, scientific programming, and numerical analysis
- Krste Asanović, Rastislav Bodik, Bryan Catanzaro, Joseph Gebis, Parry Husbands, Kurt Keutzer, Dave Patterson, William Plishker, John Shalf, Samuel Williams, Katherine Yelick + others

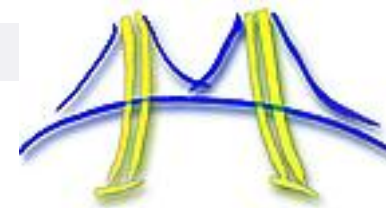
January 2007

What's the Big Idea?

- Big Idea: No (Preconceived) Big Idea!
- In past, apps considered at end of project
- Instead, work with domain experts at beginning to develop compelling applications
 - Lots of ideas now (and more to come)
- Apps determine in 3-5 yrs which ideas are big

July 2007

7 Questions for Parallelism



■ Applications:

1. What are the apps?
2. What are kernels of apps?

■ Hardware:

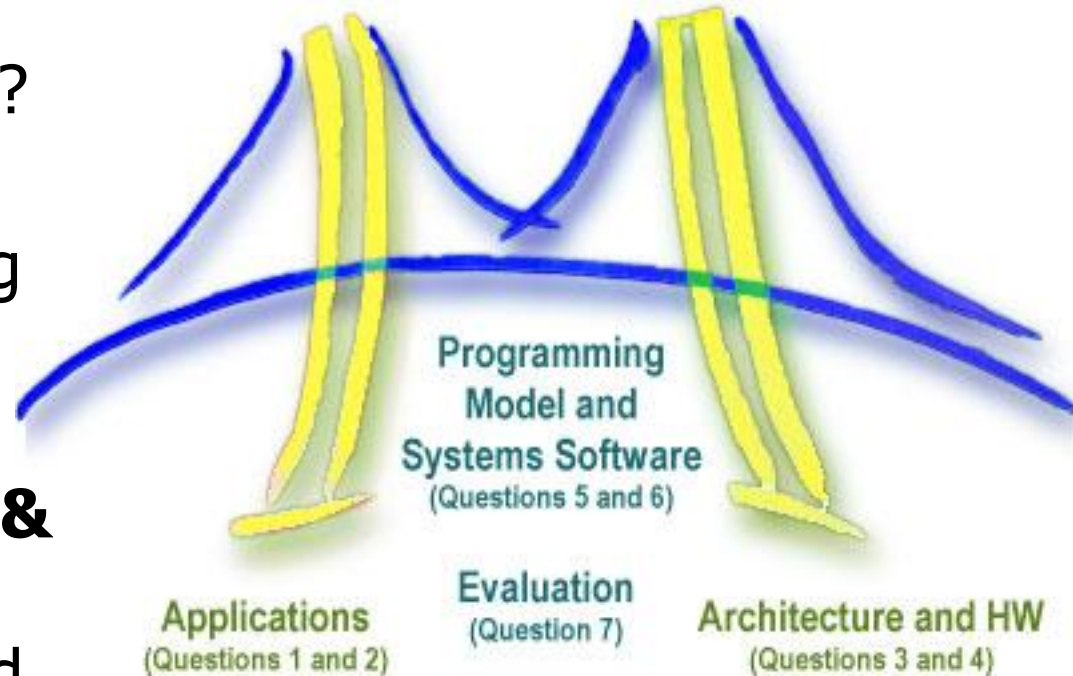
3. What are the HW building blocks?
4. How to connect them?

■ Programming Model & Systems Software:

5. How to describe apps and kernels?
6. How to program the HW?

■ Evaluation:

7. How to measure success?

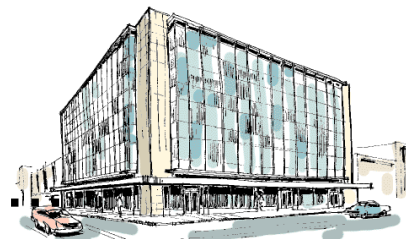


(Inspired by a view of the Golden Gate Bridge from Berkeley)

January 2007

Re-inventing Client/Server

- “The Datacenter is the Computer”
 - Building sized computers: Google, MS,
- “The Laptop/Handheld is the Computer”
 - ‘07: HP no. laptops > desktops
 - 1B+ Cell phones/yr, increasing in function
 - Otellini demoed "Universal Communicator"
 - Combination cell phone, PC and video device
- Laptop/Handheld as future client,
Datacenter as future server



April 2007

Compelling Laptop/Handheld Apps (David Wessel)

- Musicians have an insatiable appetite for computation

- More channels, instruments, more processing, more interaction!
- Latency must be low (5 ms)
- Must be reliable (No clicks)

1. Music Enhancer

- Enhanced sound delivery systems for home sound systems using large microphone and speaker arrays
- Laptop/Handheld recreate 3D sound over ear buds

2. Hearing Augmenter

- Laptop/Handheld as accelerator for hearing aide

3. Novel Instrument User Interface

- New composition and performance systems beyond keyboards
- Input device for Laptop/Handheld



Berkeley Center for New Music and Audio Technology (CNMAT) created a compact loudspeaker array: 10-inch-diameter icosahedron incorporating 120 tweeters.

April 2007

Parallel Browser

- Goal: Desktop quality browsing on handhelds
 - Enabled by 4G networks, better output devices
- Bottlenecks to parallelize
 - Parsing, Rendering, Scripting
- “SkipJax”
 - Parallel replacement for JavaScript/AJAX
 - Based on Brown’s FlapJax

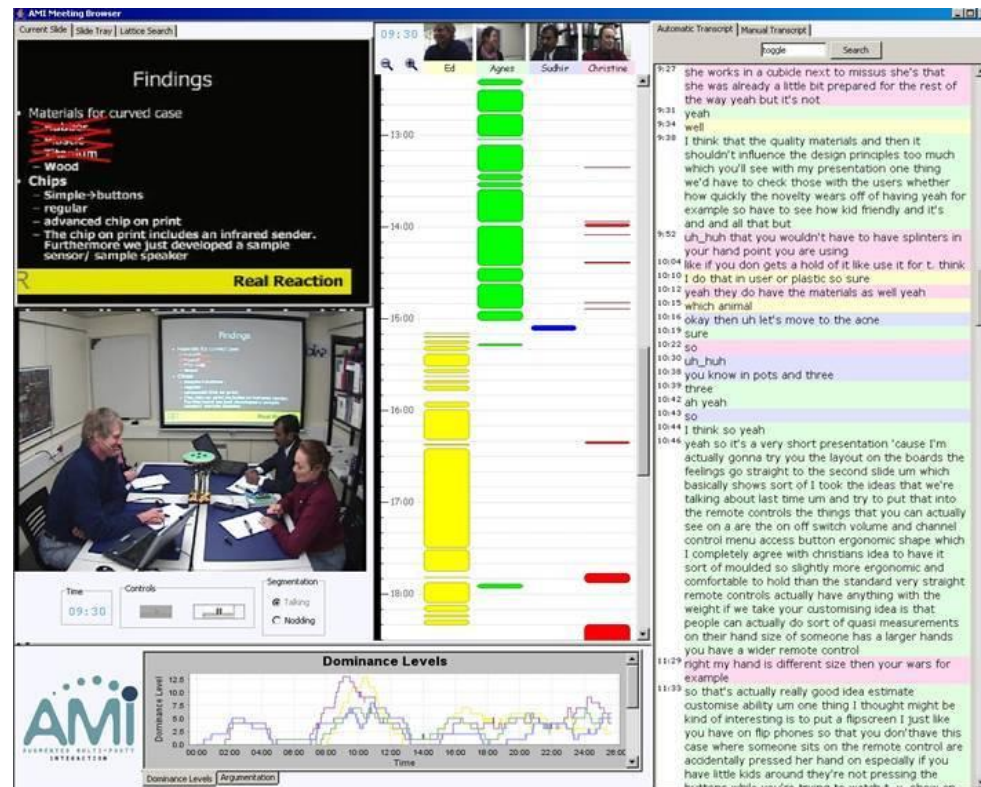
April 2007

Compelling Laptop/Handheld Apps

(Nelson Morgan)

■ Meeting Diarist

- Laptops/ Handhelds at meeting coordinate to create speaker identified, partially transcribed text diary of meeting



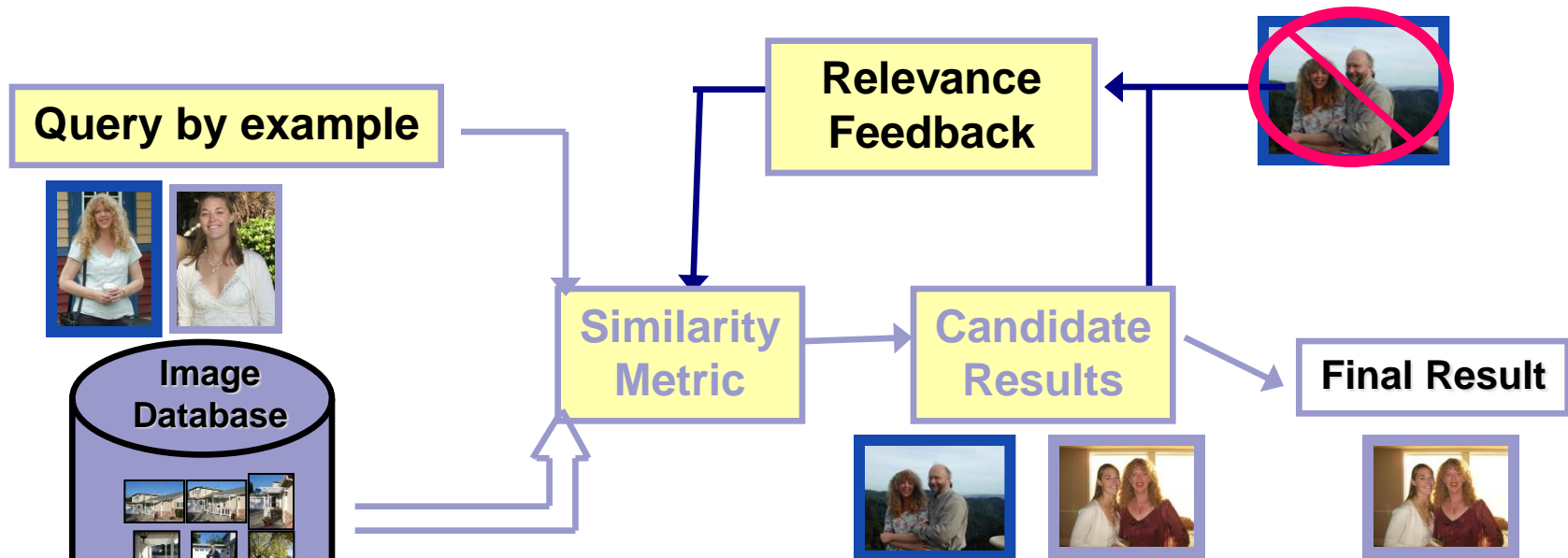
■ Teleconference speaker identifier, speech helper

- L/Hs used for teleconference, identifies who is speaking, “closed caption” hint of what being said

April 2007

Content-Based Image Retrieval

(Kurt Keutzer)



1000's of images

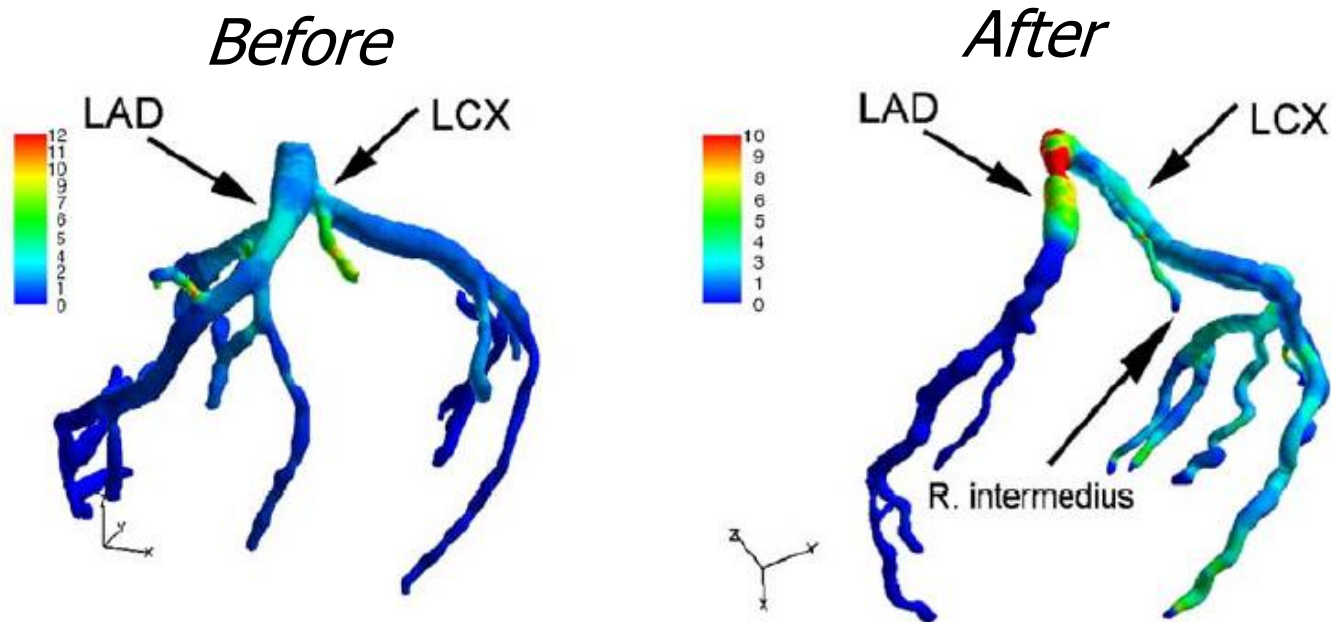
- Built around Key Characteristics of personal databases

- Very large number of pictures (>5K)
- Non-labeled images
- Many pictures of few people
- Complex pictures including people, events, places, and objects



April 2007

Coronary Artery Disease (Tony Keaveny)



- Modeling to help patient compliance?
 - 450k deaths/year, 16M w. symptom, 72M \uparrow BP
- Massively parallel, Real-time variations
 - CFD FE solid (non-linear), fluid (Newtonian), pulsatile
 - Blood pressure, activity, habitus, cholesterol

April 2007

Compelling Laptop/Handheld Apps

■ Health Coach

- Since laptop/handheld always with you, Record images of all meals, weigh plate before and after, analyze calories consumed so far
 - “What if I order a pizza for my next meal? A salad?”
- Since laptop/handheld always with you, record amount of exercise so far, show how body would look if maintain this exercise and diet pattern next 3 months
 - “What would I look like if I regularly ran less? Further?”



■ Face Recognizer/Name Whisperer

- Laptop/handheld scans faces, matches image database, whispers name in ear (relies on Content Based Image Retrieval)



April 2007

Surprisingly Accurate Prediction

- Before myfitnesspal

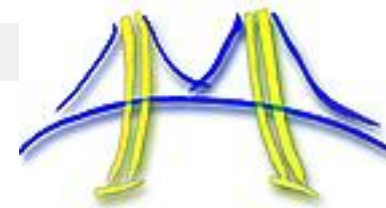


May 2013

- After myfitnesspal



Apps and Kernels



- Old CW: Since cannot know future programs, use old programs to evaluate future computers
 - e.g., SPEC2006, EEMBC
- What about parallel codes?
 - Few, tied to old models, languages, architectures, ...
- New approach: Design future computers for patterns of computation and communication important in the future
- Claim: 13 “dwarfs” are key for next decade, so design for them!
 - Representative codes may vary over time, but these dwarfs will be important for > 10 years

January 2007

Phillip Colella's "Seven dwarfs"



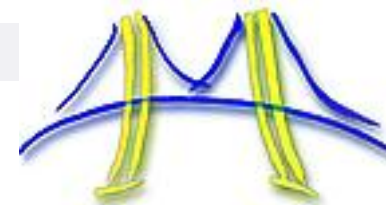
High-end simulation in the physical sciences = 7 numerical methods:

1. Structured Grids (including locally structured grids, e.g. Adaptive Mesh Refinement)
 2. Unstructured Grids
 3. Fast Fourier Transform
 4. Dense Linear Algebra
 5. Sparse Linear Algebra
 6. Particles
 7. Monte Carlo
- A dwarf is a pattern of computation and communication
 - Dwarfs are well-defined targets from algorithmic, software, and architecture standpoints

Slide from "Defining Software Requirements for Scientific Computing", Phillip Colella 2004

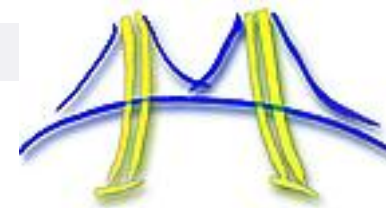
January 2007

Do dwarfs work well outside HPC?



- Examine effectiveness 7 dwarfs elsewhere
 1. Embedded Computing (EEMBC benchmark)
 2. Desktop/Server Computing (SPEC2006)
 3. Machine Learning
 - Advice from Mike Jordan and Dan Klein of UC Berkeley
 4. Games/Graphics/Vision
 5. Data Base Software
 - Advice from Jim Gray of Microsoft and Joe Hellerstein of UC
- Result: Added 7 more dwarfs, revised 2 original dwarfs, renumbered list

13 Dwarfs (so far)

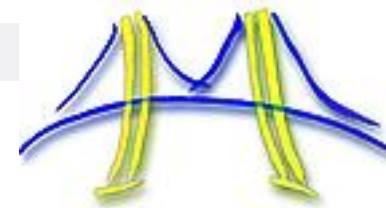


1. Dense Linear Algebra
2. Sparse Linear Algebra
3. Spectral Methods
4. N-Body Methods
5. Structured Grids
6. Unstructured Grids
7. MapReduce
8. Combinational Logic
9. Graph Traversal
10. Dynamic Programming
11. Back-track/Branch & Bound
12. Graphical Model Inference
13. Finite State Machine

- Claim is that parallel architecture, language, compiler ... that do these well will run parallel apps of future well
- Note: MapReduce is embarrassingly parallel; perhaps FSM is embarrassingly sequential?

January 2007

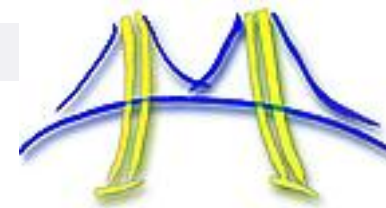
Dwarf Popularity (Red Hot → Blue Cool)



	HPC	Embed	SPEC	ML	Games	DB
1 Dense Matrix	Red	Red	Red	Red	Red	Yellow
2 Sparse Matrix	Red	Yellow	Yellow	Red	Red	Light Blue
3 Spectral (FFT)	Red	Yellow	Light Blue	Yellow	Yellow	Light Blue
4 N-Body	Red	Light Blue	Yellow	Light Blue	Yellow	Light Blue
5 Structured Grid	Red	Red	Red	Light Blue	Yellow	Light Blue
6 Unstructured	Red	Light Blue	Light Blue	Yellow	Yellow	Light Blue
7 MapReduce	Red	Light Blue	Green	Red	Light Blue	Red
8 Combinational	Light Blue	Red	Light Blue	Green	Light Blue	Green
9 Graph Traversal	Light Blue	Red	Yellow	Red	Yellow	Yellow
10 Dynamic Prog	Light Blue	Yellow	Light Blue	Red	Light Blue	Red
11 Backtrack/ B&B	Light Blue	Light Blue	Light Blue	Red	Light Blue	Yellow
12 Graphical Models	Light Blue	Light Blue	Light Blue	Red	Light Blue	Yellow
13 FSM	Light Blue	Red	Red	Yellow	Yellow	Red

January 2007

7 Questions for Parallelism



Applications:

1. What are the apps?
2. What are kernels of apps?

■ **Hardware:**

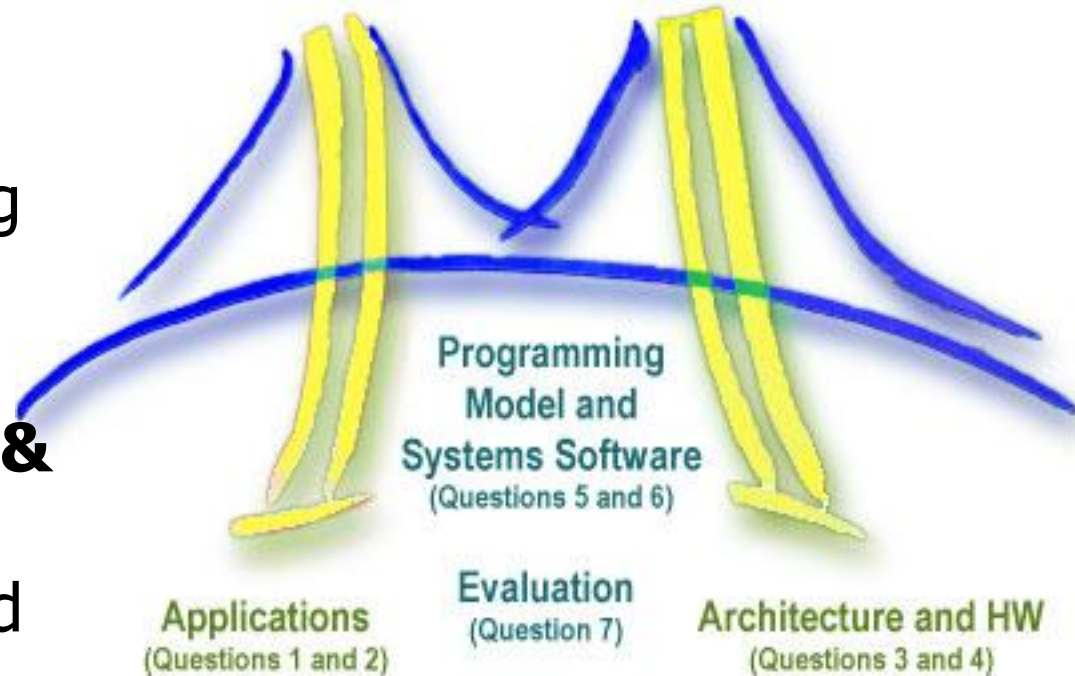
3. What are the HW building blocks?
4. How to connect them?

■ **Programming Model & Systems Software:**

5. How to describe apps and kernels?
6. How to program the HW?

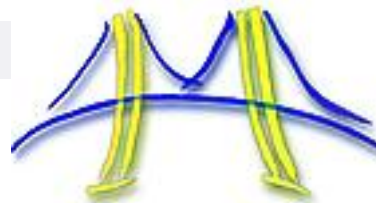
■ **Evaluation:**

7. How to measure success?



(Inspired by a view of the Golden Gate Bridge from Berkeley)

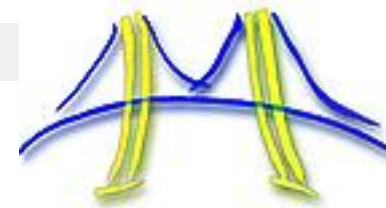
January 2007



HW Solution: Small is Beautiful

- Expect modestly pipelined (5- to 9-stage) CPUs, FPU, vector, SIMD PEs
 - Small cores not much slower than large cores
- Parallel is energy efficient path to performance: CV^2F
 - Lower threshold and supply voltages lowers energy per op
- Redundant processors can improve chip yield
 - Cisco Metro 188 CPUs + 4 spares; Sun Niagara sells 6 or 8 CPUs
- Small, regular processing elements easier to verify
- One size fits all?
 - Amdahl's Law \Rightarrow a few fast cores + many small cores?

Heterogeneous Processors?

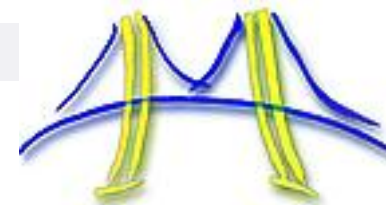


- Suppose to run the code 2X faster 1 core needs 10X resources (power, caches, ...)
- Amdahl's Law: Assume 10% time program gets no faster on manycore computer (e.g. OS)

Geneity?	Slow Cores	Fast Cores	Speedup
Homo-	100	0	9.2
Homo-	0	10	10.5
Hetero-	90	1	16.7

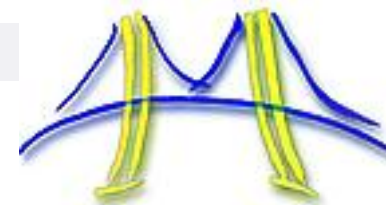
Heterogeneous same area but 1.6X to 1.8X faster

Number of Cores/Socket



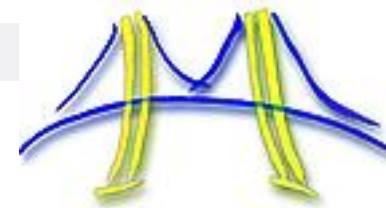
- We need revolution, not evolution
- Software or architecture alone can't fix parallel programming problem, need innovations in both
- “Multicore” 2X cores per generation: 2, 4, 8, ...
- “Manycore” 100s is highest performance per unit area, and per Watt, then 2X per generation: 128, 256, 512, 1024 ...
- **Multicore architectures & Programming Models good for 2 to 32 cores won't evolve to Manycore systems of 1000's of processors**
⇒ **Desperately need HW/SW models that work for Manycore**

Some obvious (but neglected) recommendations for hardware



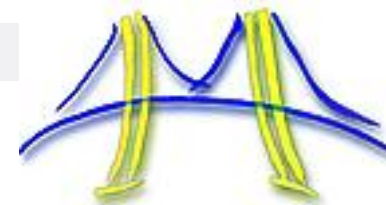
- Counters and other instrumentation more important than in the past
 - Needed for Feedback directed applications
 - Since energy is limit, include energy counters as well as performance counters
- Include counters that work!
 - In past low priority, so ship even if counters broken, or don't slow processor to measure it
 - If can't measure feature, won't use it effectively
- Don't include features that significantly affect performance or energy if programmers cannot accurately measure their impact

How to Connect Processors?



- **Topic wide open! (HW/SW innovations ASAP!)**
- 13 Dwarfs to gain insight into Networks On a Chip
 - Sparse connectivity for dwarfs; crossbar is overkill
 - No single best topology
- A Bandwidth-oriented network for data
 - Most point-to-point message are large and BW bound
- Separate Latency-oriented network for collectives
 - Given BW improves $> (\text{latency improvement})^2$
 - E.g., Thinking Machines CM-5, Cray T3D, IBM BlueGene/L&P
- Virtual circuit switch??
- Synchronization??
 - Transactional memory, full-empty bits, barriers???
 - Is cache coherency all we need to coordinate cores?

7 Questions for Parallelism



Applications:

1. What are the apps?
2. What are kernels of apps?

■ Hardware:

3. What are the HW building blocks?

4. How to connect them?

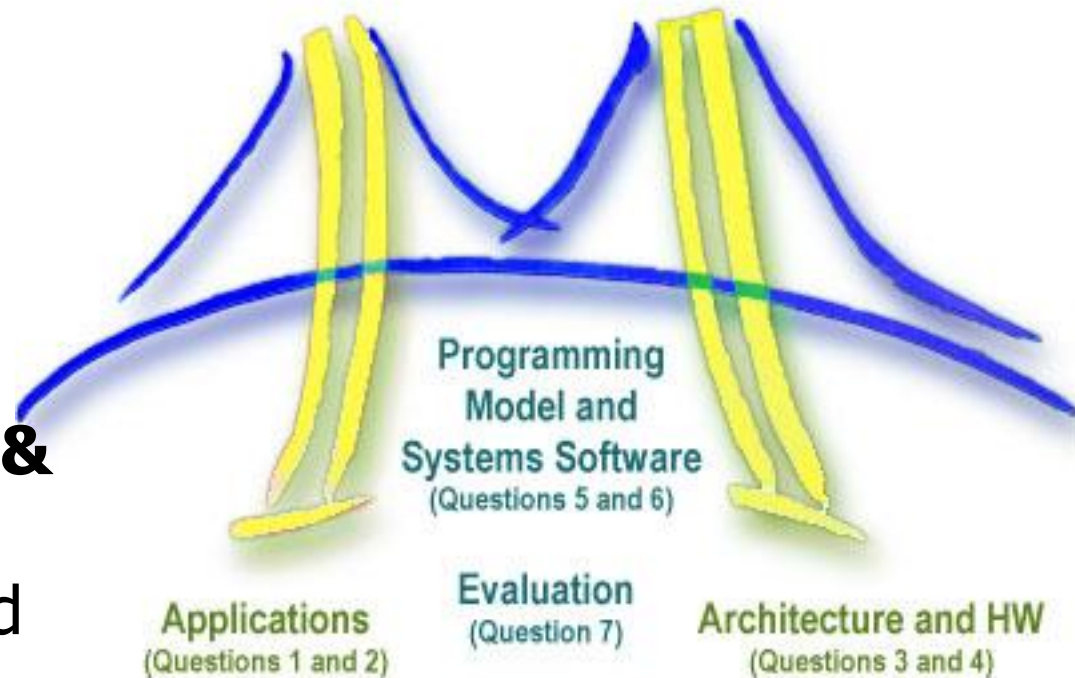
■ **Programming Model & Systems Software:**

5. How to describe apps and kernels?

6. How to program the HW?

■ **Evaluation:**

7. How to measure success?



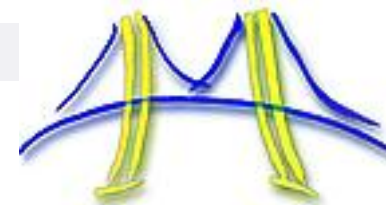
(Inspired by a view of the Golden Gate Bridge from Berkeley)

January 2007

Programming Model

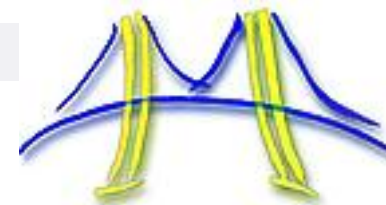
- Programming model must allow programmer to balance competing goals of *productivity* and *implementation efficiency*
 - Biggest challenge facing manycore systems
- ⇒ **Programming Model high priority**
- Past foci of parallel Programming Models:
 1. Hardware-centric (e.g., C-variants)
 2. Application-centric (e.g., MatLab)
 3. Formalism-centric (e.g., Sisal)

21st Century Code Generation



- Takes a decade for compiler innovations to show up in production compilers?
- New approach: “**Auto-tuners**” 1st run variations of program on computer to find best combinations of optimizations (blocking, padding, ...) and algorithms, then produce C code to be compiled for *that* computer
 - E.g., PHiPAC (BLAS), Atlas (BLAS), Sparsity (Sparse linear algebra), Spiral (DSP), FFT-W
 - Can achieve 10X over conventional compiler
- One Auto-tuner per kernel or dwarf?
 - Exist for Dense Linear Algebra, Sparse Linear Algebra, Spectral

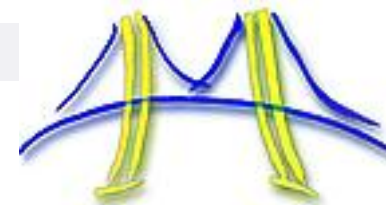
Deconstructing Operating Systems



- Resurgence of interest in virtual machines
 - Traditional OSes brittle & too large (AIX GBs DRAM)
 - VM monitor thin SW layer btw guest OS and HW
- Advantages
 - Security via isolation
 - VMs move from failing processor
- Mendel Rosenblum: future of OSes could be libraries where only functions needed are linked into app, on top of thin VMM layer providing protection and sharing of resources
 - Everywhere, but great match to 1000s of processors

January 2007

7 Questions for Parallelism



- **Applications:**

1. What are the apps?
2. What are kernels of apps?

- **Hardware:**

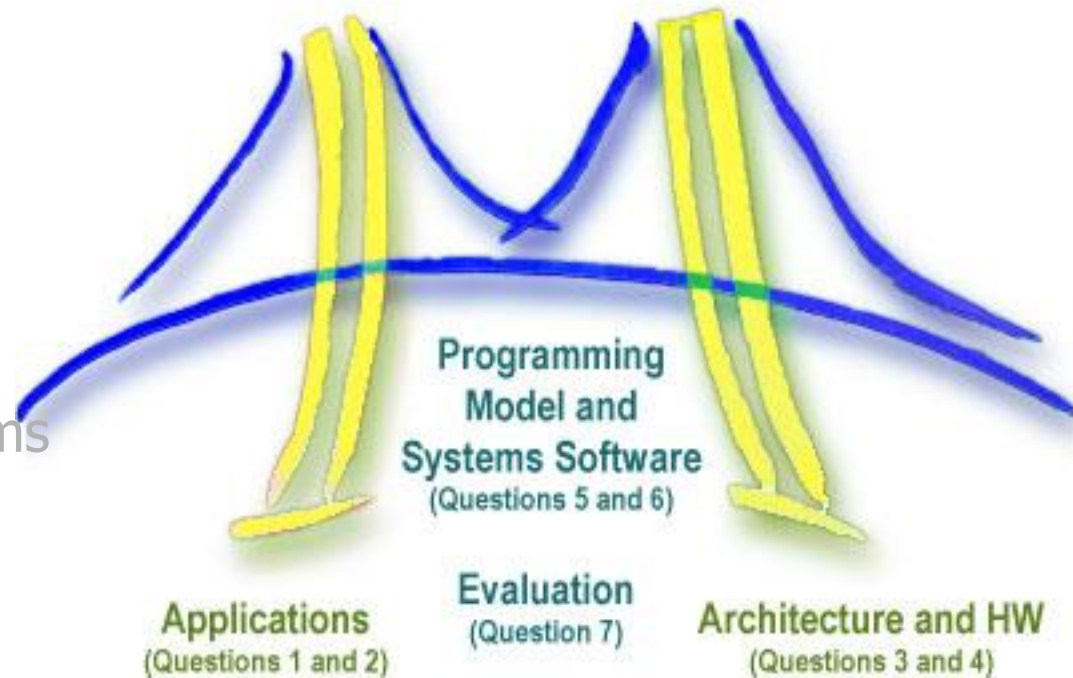
3. What are the HW building blocks?
4. How to connect them?

Programming Model & Systems Software:

5. How to describe apps and kernels?
6. How to program the HW?

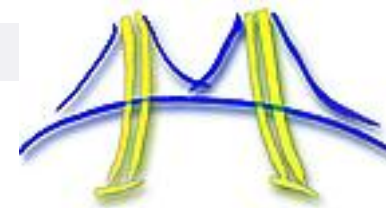
- **Evaluation:**

7. How to measure success?



(Inspired by a view of the Golden Gate Bridge from Berkeley)

How to measure success?



- Easy to write programs that execute efficiently on manycore computing systems
 1. Maximizing programmer productivity
 2. Maximizing application performance and energy efficiency
- Challenges
 - Conventional Serial Performance Issues
 - Minimizing Remote Accesses
 - Balancing Load
 - Granularity of Data Movement and Synchronization



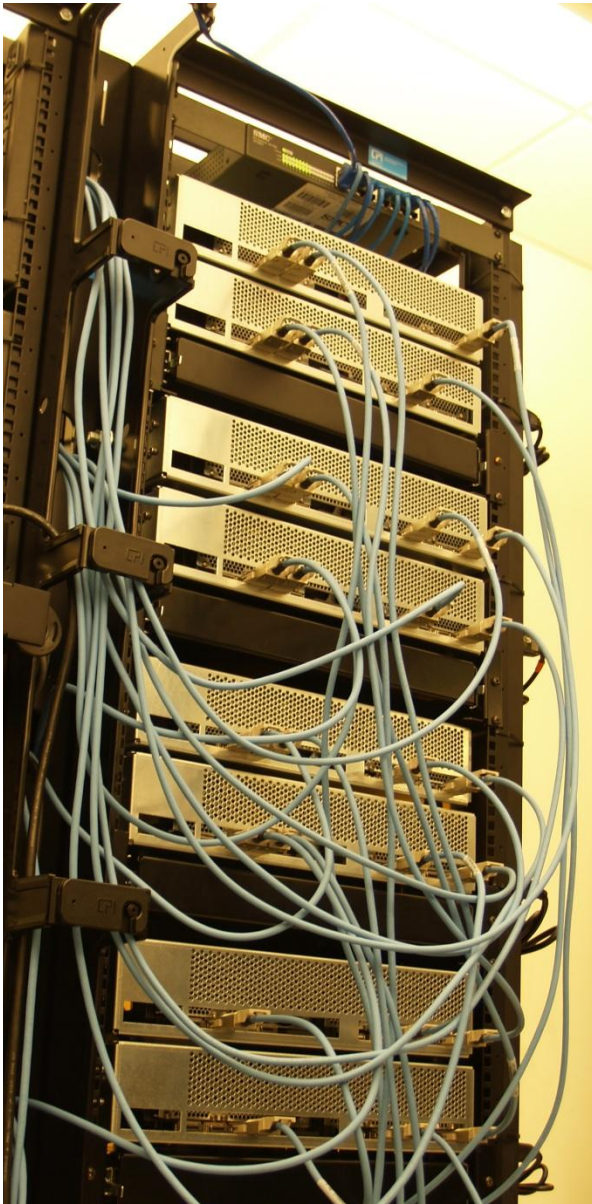
Problems with “Manycore” Sea Change

1. Algorithms, Programming Languages, Compilers, Operating Systems, Architectures, Libraries, ... not ready for 1000 CPUs / chip
2. \approx Only companies can build HW, and it takes years
3. Software people don't start working hard until hardware arrives
 - 3 months after HW arrives, SW people list everything that must be fixed, then we all wait 4 years for next iteration of HW/SW
4. How get 1000 CPU systems in hands of researchers to innovate in timely fashion on in algorithms, compilers, languages, OS, architectures, ... ?
5. Can avoid waiting years between HW/SW iterations?

Build Academic Manycore from FPGAs

- As ≈ 16 CPUs will fit in Field Programmable Gate Array (FPGA), 1000-CPU system from ≈ 64 FPGAs?
 - 8 32-bit simple “soft core” RISC at 100MHz in 2004 (Virtex-II)
 - FPGA generations every 1.5 yrs; $\approx 2X$ CPUs, $\approx 1.2X$ clock rate
- HW research community does logic design (“gate shareware”) to create out-of-the-box, Manycore
 - E.g., 1000 processor, standard ISA binary-compatible, 64-bit, cache-coherent supercomputer @ ≈ 150 MHz/CPU in 2007
 - RAMPants: 10 faculty at Berkeley, CMU, MIT, Stanford, Texas, and Washington
- “Research Accelerator for Multiple Processors” as a vehicle to attract many to parallel challenge



256 CPU Message Passing/RAMP Blue



- 8 MicroBlaze cores / FPGA
- 8 BEE2 modules (32 “user” FPGAs) x 4 FPGAs/module = 256 cores @ 100MHz
 - \$10k/board
- Full star-connection between modules
- It works; runs NAS benchmarks in UPC
- Cores are softcore MicroBlazes (32-bit Xilinx RISC)
- Schultz, Krasnov, Wawrzynek at Berkeley
January 2007

Change directions of research funding?



Historically:
Get leading
experts per
discipline
(across US)
working
together
to work on
parallelism

		CMU		Stanford	...
Application					
Language					
Compiler					
Libraries					
Networks					
Architecture					
Hardware					
CAD					

January 2007

Change directions of research funding?

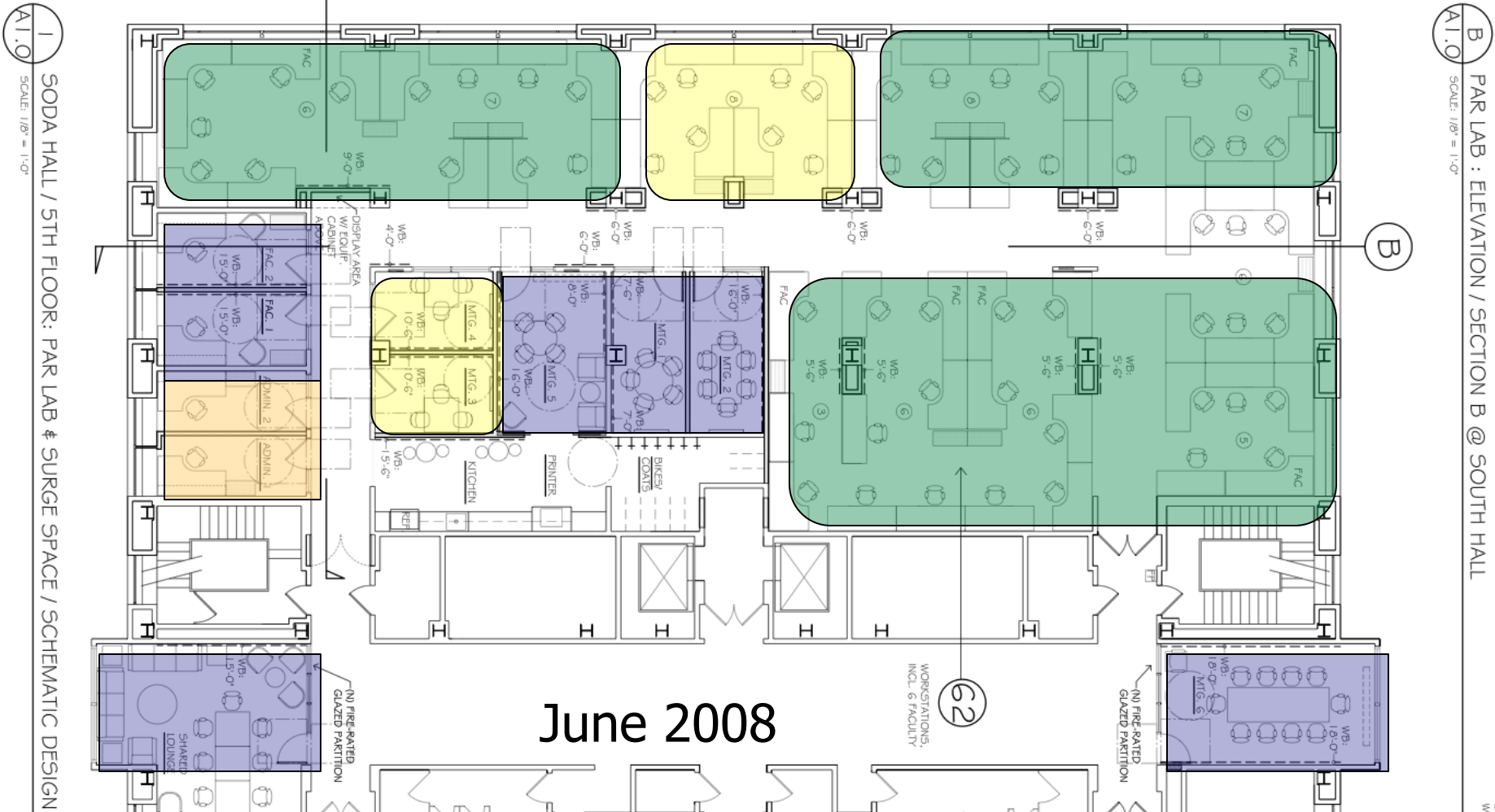
To increase cross-disciplinary bandwidth, get experts **per site** working together on parallelism

		CMU		Stanford	...
Application					
Language					
Compiler					
Libraries					
Networks					
Architecture					
Hardware					
CAD					

January 2007

Physical Par Lab: Maximizing Communication & Concentration

Staff Meeting Rooms Faculty Students/Postocs



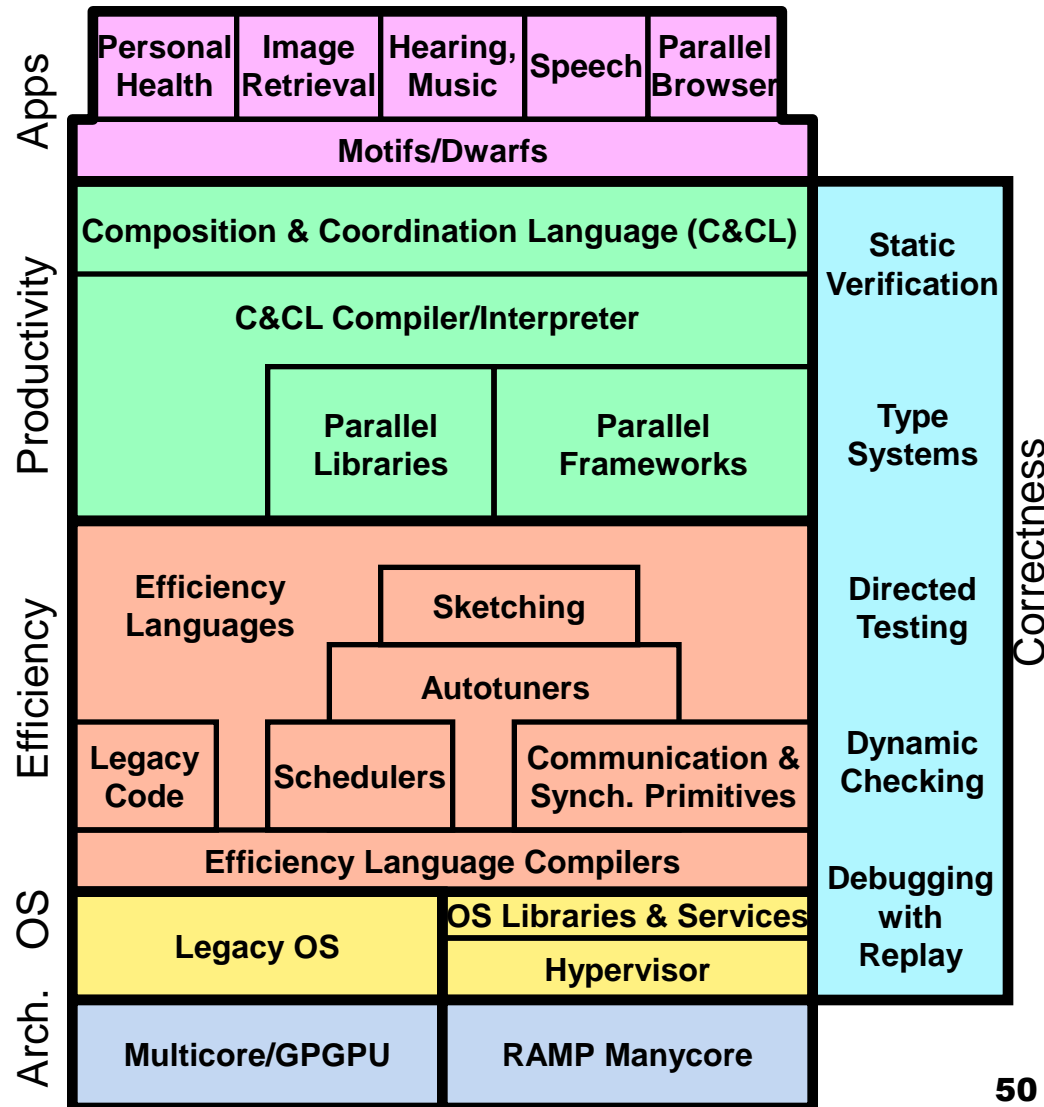
June 2008

Summary: A Berkeley View 2.0

- Whole IT industry has bet its future on parallelism (!)
 - Recruit best minds to help?
- Try Apps-Driven vs. CS Solution-Driven Research
- Motifs/dwarfs as lingua franca, anti-benchmarks...
- Efficiency layer for $\approx 10\%$ today's programmers
- Productivity layer for $\approx 90\%$ today's programmers
- C&C language to help compose and coordinate
- Autotuners vs. Parallelizing Compilers
- OS & HW: Composable Primitives vs. Solutions

April 2007

Easy to write correct programs that run efficiently and scale up on manycore



Where to go from here?

- What bold new applications will manycore enable?
- Can we design architectures that make parallel programming easier?
- Can we develop highly-productive programming models that harness the performance of manycore?
- Berkeley is one ideal place to do the cross-disciplinary research needed to save the IT industry's desperate bet on parallelism

A black and white photograph of a clock face. The clock has a large outer dial and a smaller inner dial. The numbers 1 through 12 are visible on the outer dial. The text "JAN 2007" is overlaid in large, bold, white capital letters across the center of the clock face. The clock hands are visible, and the overall image has a vintage, slightly grainy appearance.

JAN 2007