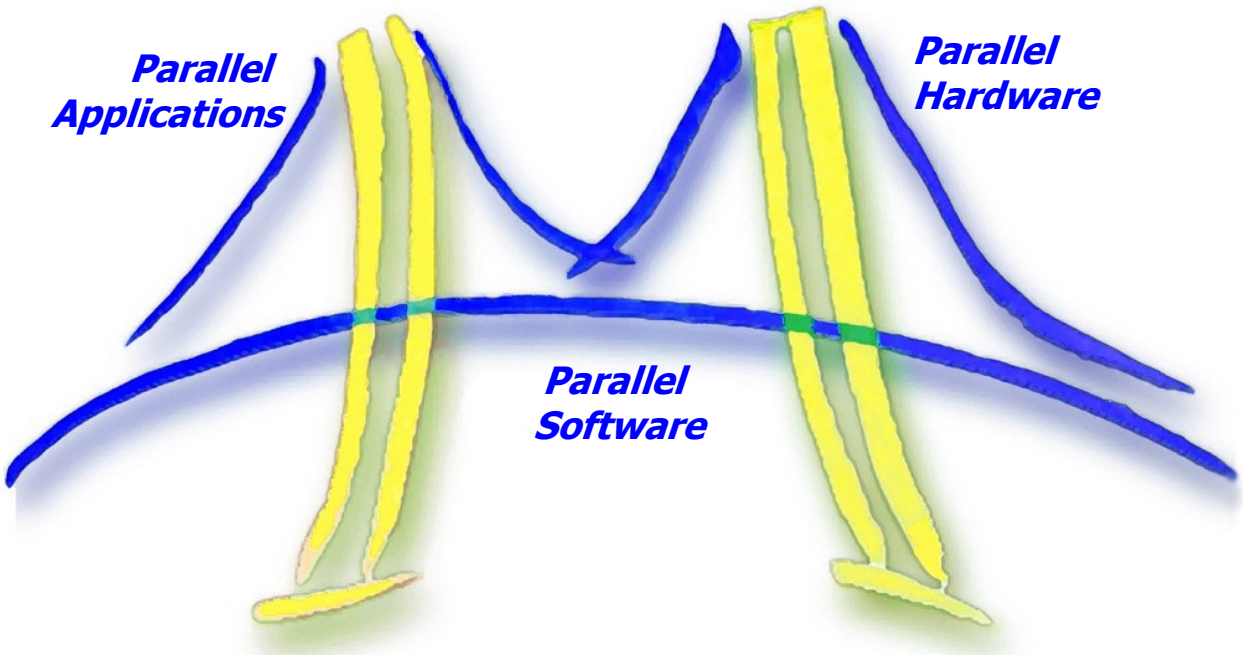
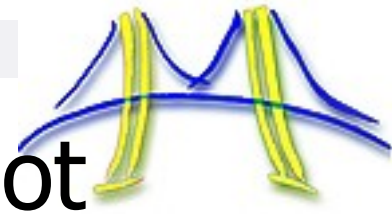


The Parallel Computing Landscape: A View from Berkeley 2.0

Krste Asanovic, Ras Bodik, Jim Demmel, Tony Keaveny,
Kurt Keutzer, John Kubiatowicz, Edward Lee, Nelson Morgan,
George Necula, Dave Patterson, Koushik Sen,
John Wawrzynek, David Wessel, and Kathy Yelick

November, 2007

A Parallel Revolution, Ready or Not



- Embedded: per product ASIC to programmable platforms ⇒ Multicore chip most competitive path
  - Amortize design costs + Reduce design risk + Flexible platforms
- PC, Server: Power Wall + Memory Wall = Brick Wall
  - ⇒ End of way built microprocessors for last 40 years

⇒ New Moore's Law is 2X processors ("cores") per chip every technology generation, but same clock rate

- "This shift toward increasing parallelism is not a triumphant stride forward based on breakthroughs …; instead, this … **is actually a retreat from even greater challenges that thwart efficient silicon implementation of traditional solutions**."

  The Parallel Computing Landscape: A Berkeley View, Dec 2006

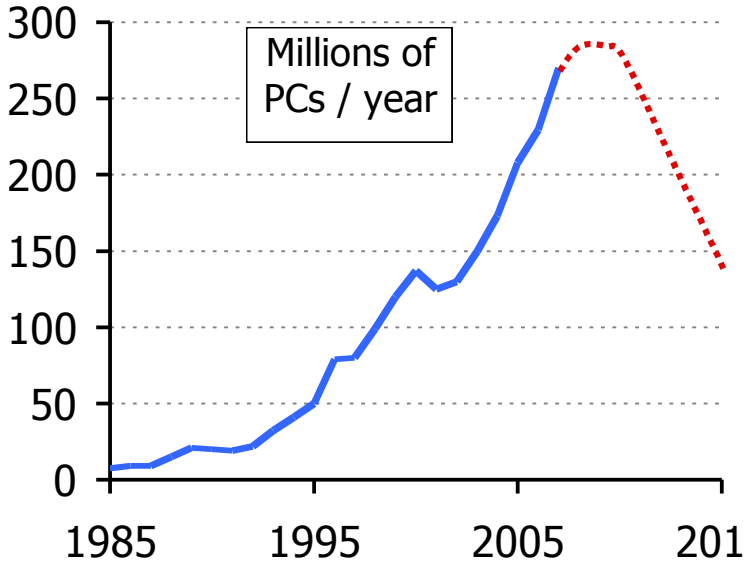
- Sea change for HW & SW industries since changing the model of programming and debugging

P.S. Parallel Revolution May Fail

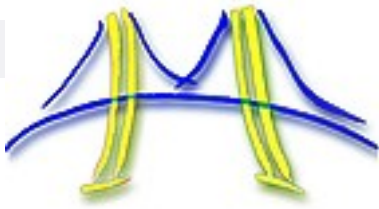
- John Hennessy, President, Stanford University, 1/07:
  "…when we start talking about parallelism and ease of use of truly parallel computers, we're talking about a problem that's as hard as any that computer science has faced. … I would be panicked if I were in industry."

"A Conversation with Hennessy & Patterson," *ACM Queue Magazine*, 4:10, 1/07.

- 100% failure rate of Parallel Computer Companies
  - Convex, Encore, MasPar, NCUBE, Kendall Square Research, Sequent, (Silicon Graphics), Transputer, 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



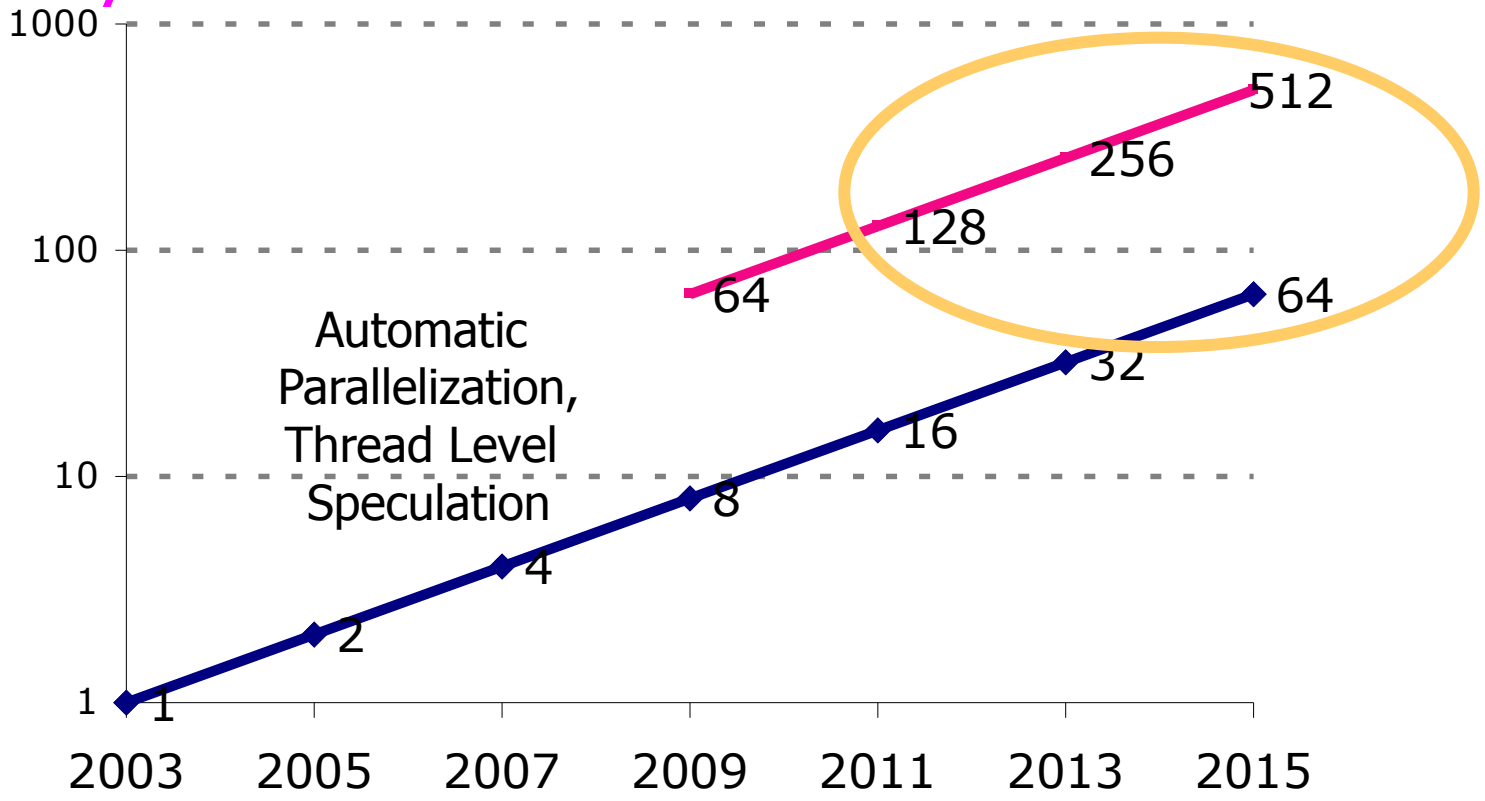
Need a Fresh Approach to Parallelism



- Berkeley researchers from many backgrounds meeting since Feb. 2005 to discuss parallelism
  - Krste Asanovic, Ras Bodik, Jim Demmel, Kurt Keutzer, John Kubiatowicz, Edward Lee, George Necula, Dave Patterson, Koushik Sen, John Shalf, John Wawrzynek, Kathy Yelick, …
  - Circuit design, computer architecture, massively parallel computing, computer-aided design, embedded hardware and software, programming languages, compilers, scientific programming, and numerical analysis
- Tried to learn from successes in high performance computing (LBNL) and parallel embedded (BWRC)
- Led to "Berkeley View" Tech. Report and new Parallel Computing Laboratory ("Par Lab")
- Goal: Productive, Efficient, Correct Programming of 100+ cores & scale as double cores every 2 years (!)

Why Target 100+ Cores?

- 5-year research program aim 8+ years out
- Multicore: 2X / 2 yrs ⇒ ≈ 64 cores in 8 years
- Manycore: 8X to 16X multicore



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
  - Apple iPhone

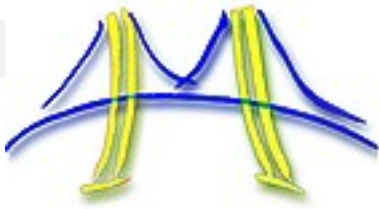
- Laptop/Handheld as future client, Datacenter as future server

Laptop/Handheld Reality

- Use with large displays at home or office
- What % time disconnected? 10%? 30% 50%?
  - Disconnectedness due to Economics
    - Cell towers and support system expensive to maintain ⇒ charge for use to recover costs ⇒ costs to communicate
    - Policy varies, but most countries allow wireless investment where make most money ⇒ Cities well covered ⇒ Rural areas will never be covered
  - Disconnectedness due to Technology
    - No coverage deep inside buildings
    - Satellite communication uses up batteries

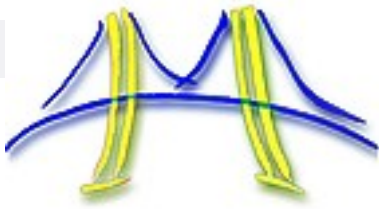
⇒ Need computation & storage in Laptop/Handheld

Try Application Driven Research?



- Conventional Wisdom in CS Research
  - Users don't know what they want
  - Computer Scientists solve individual parallel problems with clever language feature (e.g., futures), new compiler pass, or novel hardware widget (e.g., SIMD)
  - Approach: Push (foist?) CS nuggets/solutions on users
  - Problem: Stupid users don't learn/use proper solution
- Another Approach
  - Work with domain experts developing compelling apps
  - Provide HW/SW infrastructure necessary to build, compose, and understand parallel software written in multiple languages
  - Research guided by commonly recurring patterns actually observed while developing compelling app

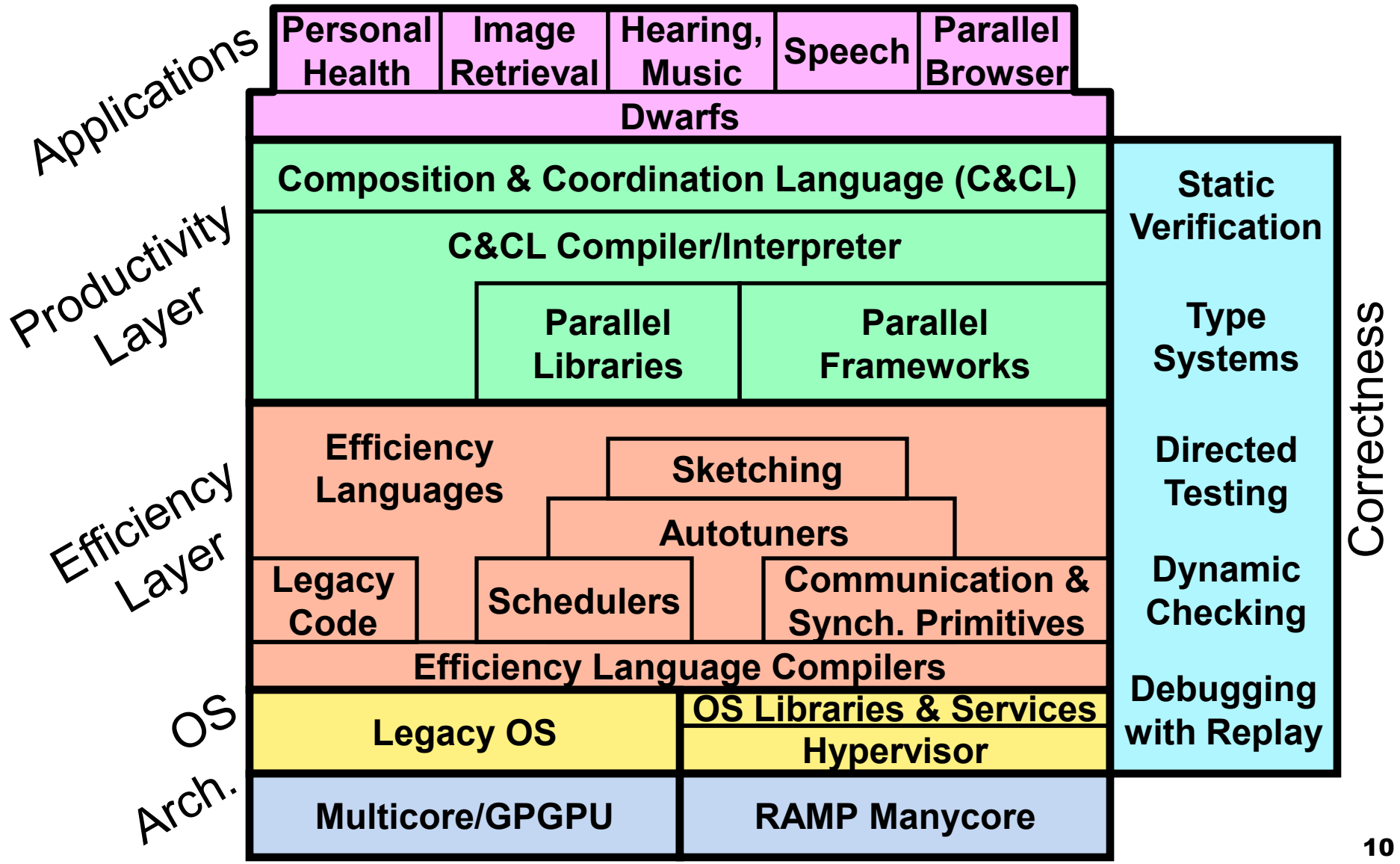
4 Themes of View 2.0/ Par Lab



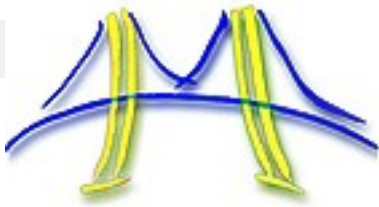
1. Applications
   - Compelling apps drive top-down research agenda
2. Identify Common Computational Bottlenecks
   - Breaking through disciplinary boundaries
3. Developing Parallel Software with Productivity, Efficiency, and Correctness
   - 2 Layers + Coordination & Composition Language + Autotuning
4. OS and Architecture
   - Composable primitives, not packaged solutions
   - Deconstruction, Fast barrier synchronization, Partitions

Par Lab Research Overview

Easy to write correct programs that run efficiently on manycore



Theme 1. Applications. What are the problems? (Jim Demmel)

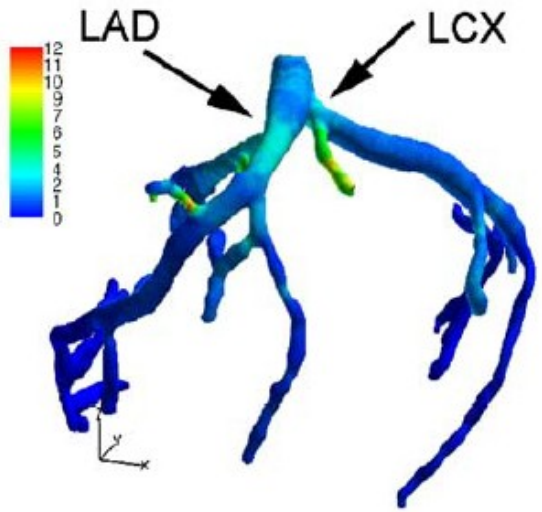


- "Who needs 100 cores to run M/S Word?"
  - Need compelling apps that use 100s of cores
- How did we pick applications?

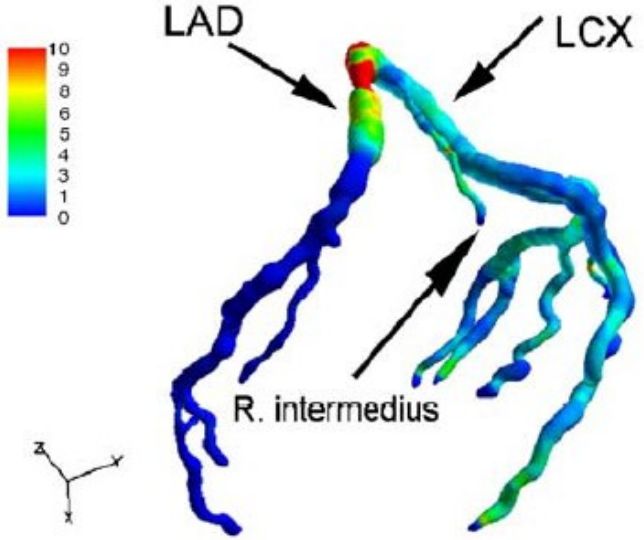
1. Enthusiastic expert application partner, leader in field, promise to help design, use, evaluate our technology
2. Compelling in terms of likely market or social impact, with short term feasibility and longer term potential
3. Requires significant speed-up, or a smaller, more efficient platform to work as intended
4. As a whole, applications cover the most important
   - Platforms (handheld, laptop, games)
   - Markets (consumer, business, health)

Coronary Artery Disease (Tony Keaveny)

Before



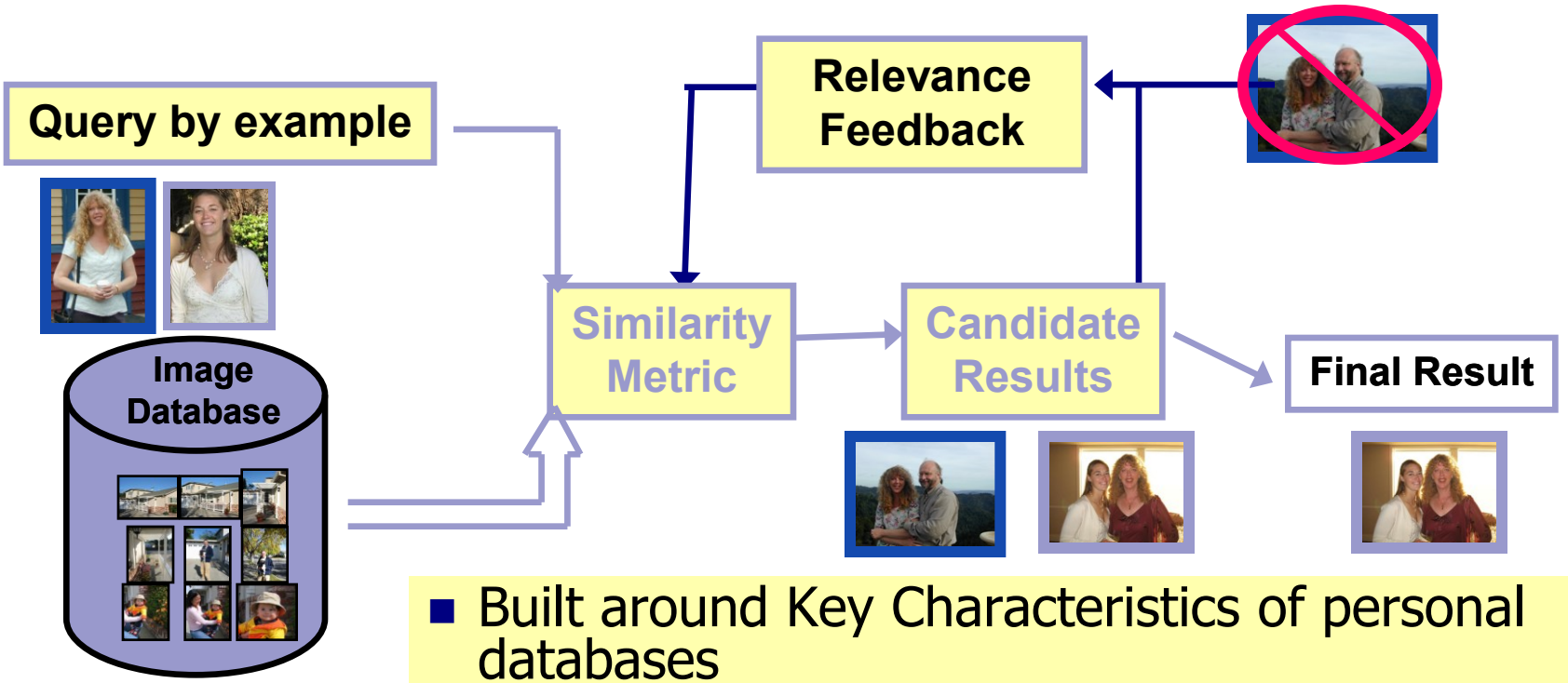
After



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

Content-Based Image Retrieval

(Kurt Keutzer)

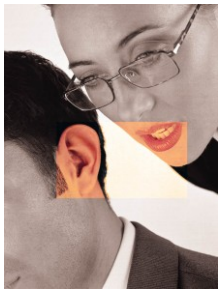


1000's of images

- 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

Compelling Laptop/Handheld Apps

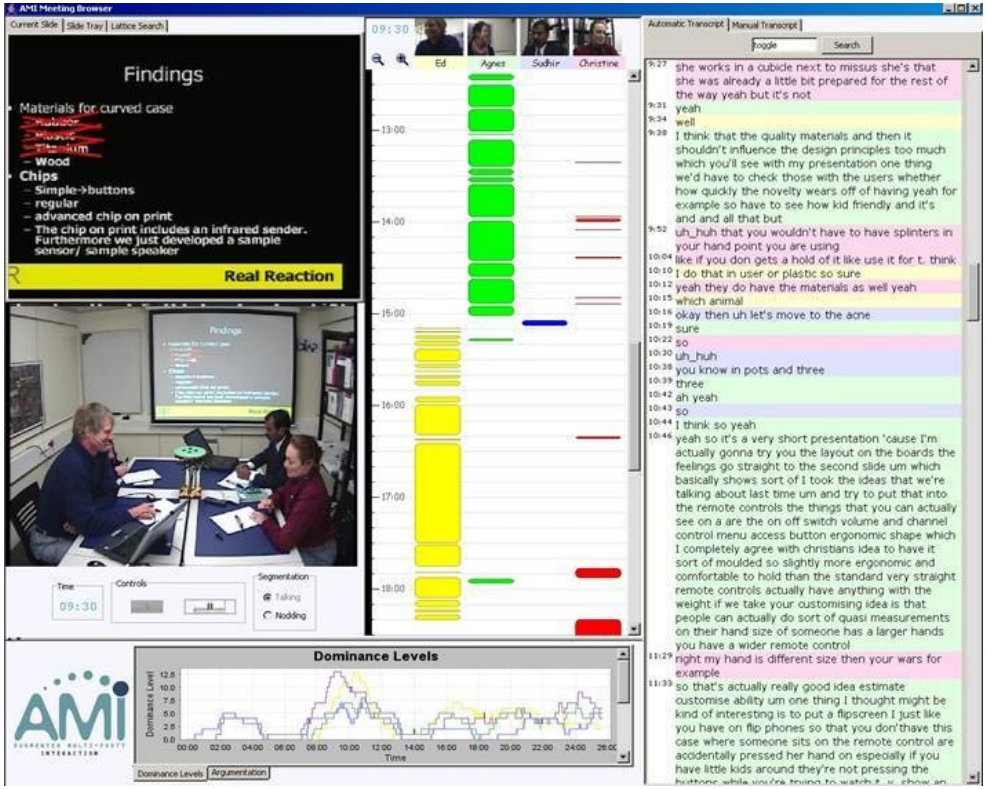
- Health Coach
  - Since laptop/handheld always with you, Record images of all meals, weigh plate before and after, analyze calories consumed so far
    - "What if I order a pizza for my next meal? A salad?"
  - Since laptop/handheld always with you, record amount of exercise so far, show how body would 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)



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

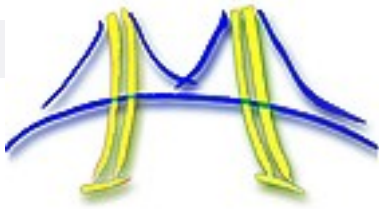
Compelling Laptop/Handheld Apps (David Wessel)

- 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

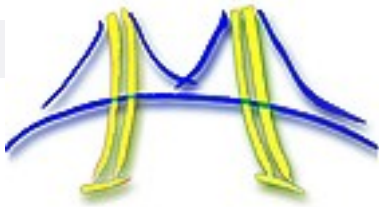
Berkeley Center for New Music and Audio Technology (CNMAT) created a compact loudspeaker array: 10-inch-diameter icosahedron incorporating 120 tweeters.

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

Theme 2. Use computational bottlenecks instead of conventional benchmarks?



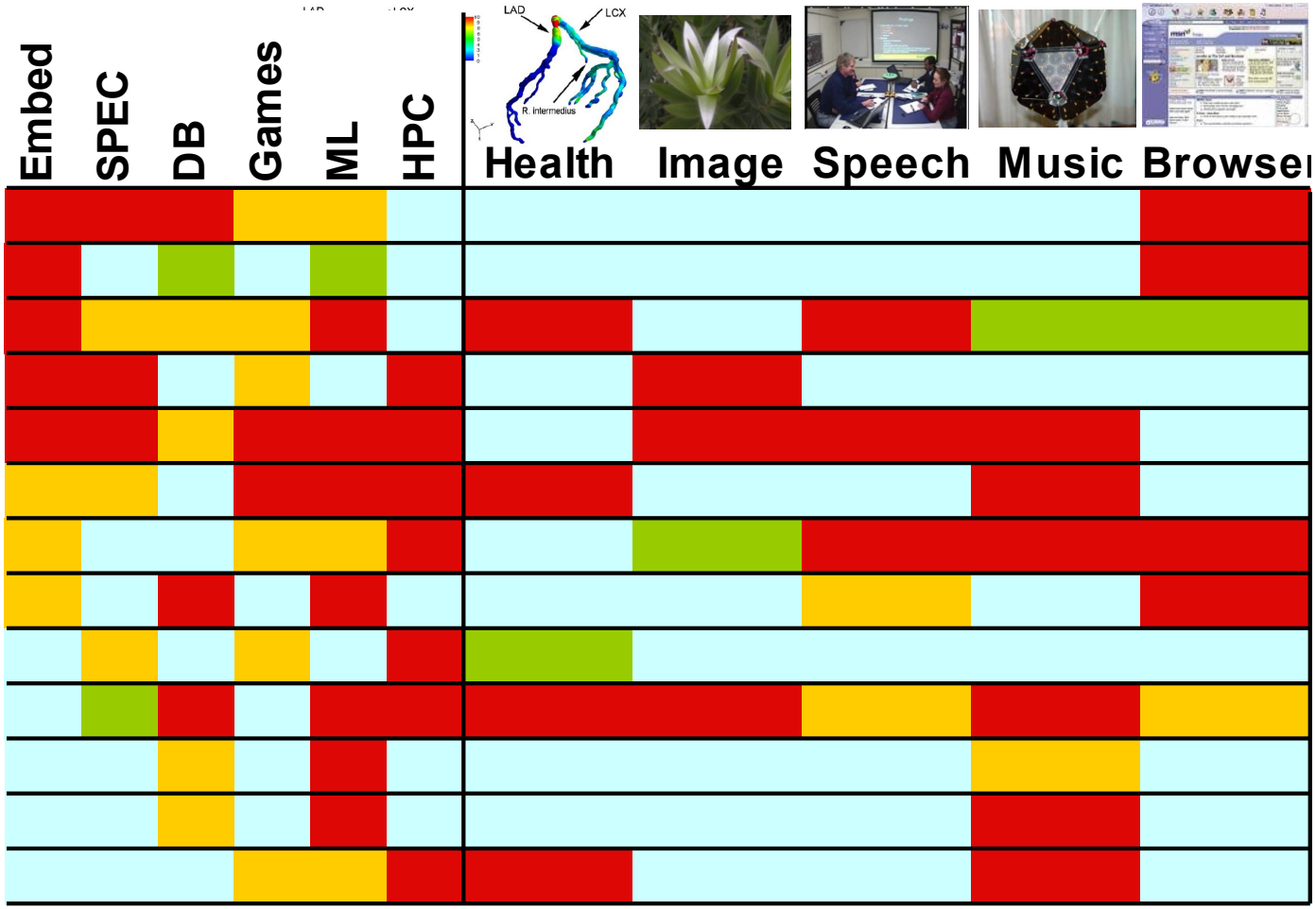
- How invent parallel systems of future when tied to old code, programming models, CPUs of the past?
- Look for common patterns of communication and computation

1. Embedded Computing (42 EEMBC benchmarks)
2. Desktop/Server Computing (28 SPEC2006)
3. Data Base / Text Mining Software
4. Games/Graphics/Vision
5. Machine Learning
6. High Performance Computing (Original "7 Dwarfs")

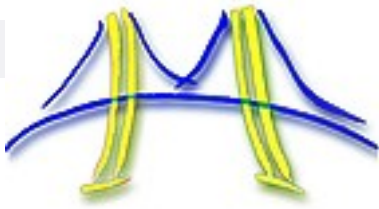
- Result: 13 "Dwarfs"

Dwarf Popularity (Red Hot → Blue Cool)

- How do compelling apps relate to 13 dwarfs?

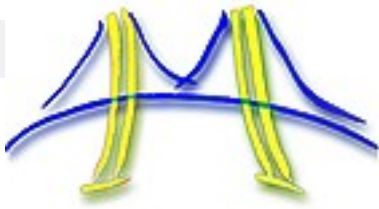


What about I/O and dwarfs?



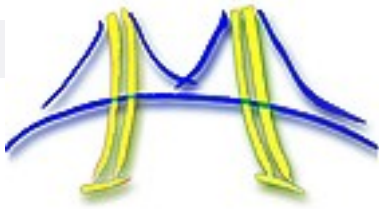
- High speed serial lines
  ⇒ many parallel I/O channels per chip
- Storage will be much faster
  - "Flash is disk, disk is tape"
    - Disk ⇒ Flash means 1000x improvement in latency (≈10 millisecond to ≈10 microsecond)
    - At least for laptop/handheld
    - Camera, iPod, cell phone, laptop ⇒ $ to improve flash memory
    - New technologies even more promising:
      Phase Change RAM denser, faster write, longer wear-out, …
- Network will be much faster
  - 10 Gbit/s copper is standard; 100 Gbit/sec is coming
  - Mutiple serial channels to multiply bandwidth
  - Processor speed limited effective network BW in past
    ⇒ Can use up cores to handle TCP/IP at 10 Gbit/s

4 Valuable Roles of Dwarfs



1. "Anti-benchmarks"
   - Dwarfs not tied to code or language artifacts $\Rightarrow$ encourage innovation in algorithms, languages, data structures, and/or hardware
2. Universal, understandable vocabulary
   - To talk across disciplinary boundaries
3. Bootstrapping: Parallelize parallel research
   - Allow analysis of HW & SW design without waiting years for full apps
4. Targets for libraries (see later)

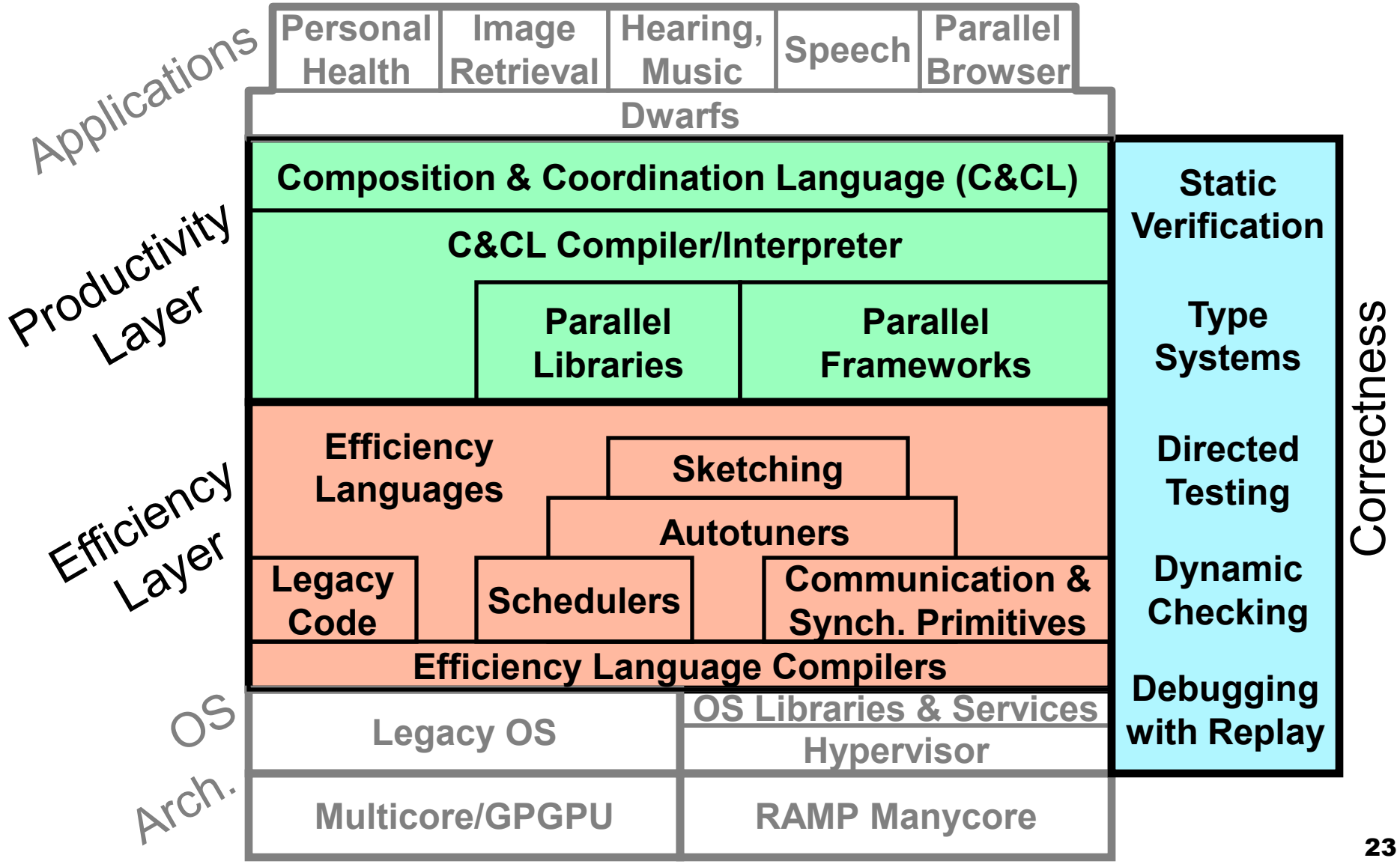
Themes 1 and 2 Summary



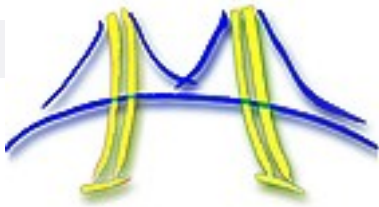
- Application-Driven Research vs. CS Solution-Driven Research
- Drill down on (initially) 5 app areas to guide research agenda
- Dwarfs to represent broader set of apps to guide research agenda
- Dwarfs help break through traditional interfaces
  - Benchmarking, multidisciplinary conversations, parallelizing parallel research, and target for libraries

Par Lab Research Overview

Easy to write correct programs that run efficiently on manycore



Theme 3: Developing Parallel SW (Kurt Keutzer and Kathy Yelick)



- **Observation: Use Dwarfs. Dwarfs are of 2 types**

Algorithms in the dwarfs can either be implemented as:

- Compact parallel computations within a traditional ***library***
- Compute/communicate pattern implemented as ***framework***

Libraries

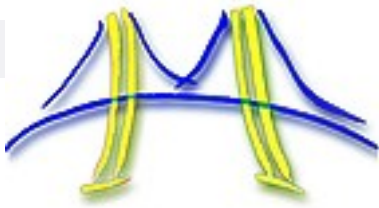
- Dense matrices
- Sparse matrices
- Spectral
- Combinational
- Finite state machines

Patterns/Frameworks

- MapReduce
- Graph traversal, graphical models
- Dynamic programming
- Backtracking/B&B
- N-Body
- (Un) Structured Grid

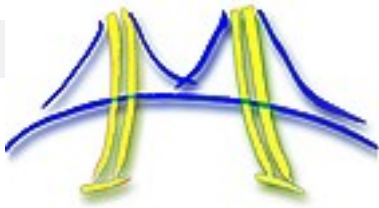
- Computations may be viewed at multiple levels: e.g., an FFT library may be built by instantiating a Map-Reduce framework, mapping 1D FFTs and then transposing (generalized reduce)

Developing Parallel Software



- 2 types of programmers ⇒ 2 layers
- **Efficiency Layer** (10% of today's programmers)
  - Expert programmers build Frameworks & Libraries, Hypervisors, …
  - "Bare metal" efficiency possible at Efficiency Layer
- **Productivity Layer** (90% of today's programmers)
  - Domain experts / Naïve programmers productively build parallel apps using frameworks & libraries
  - Frameworks & libraries composed to form app frameworks
- Effective composition techniques allows the efficiency programmers to be highly leveraged ⇒
  Create language for Composition and Coordination (C&C)

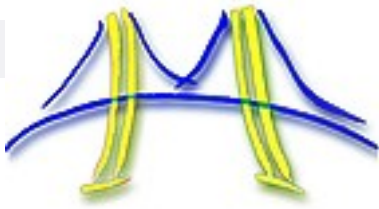
C & C Language Requirements



Applications specify C&C language requirements:

- Constructs for creating application frameworks
- Primitive parallelism constructs:
  - Data parallelism
  - Divide-and-conquer parallelism
  - Event-driven execution
- Constructs for composing programming frameworks:
  - Frameworks require independence
  - Independence is proven at instantiation with a variety of techniques
- Needs to have low runtime overhead and ability to measure and control performance

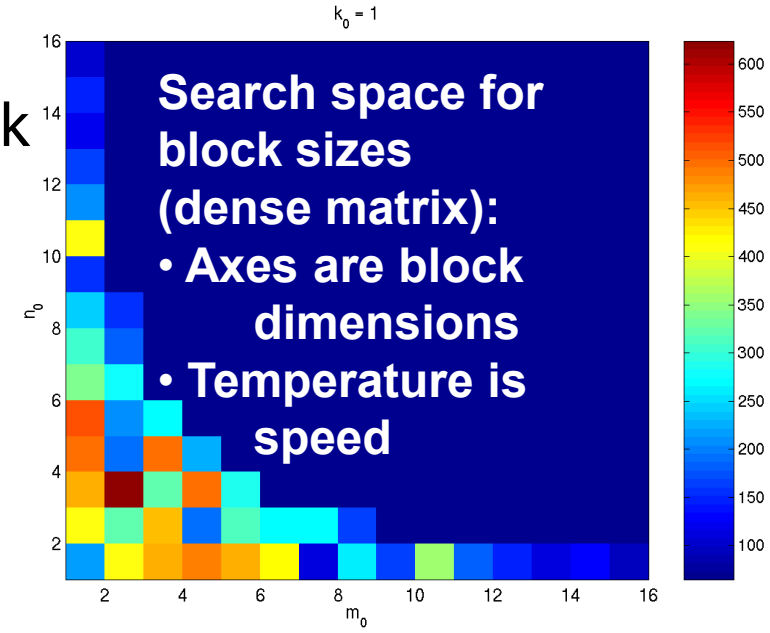
Ensuring Correctness



- Productivity Layer
  - Enforce independence of tasks using decomposition (partitioning) and copying operators
  - Goal: Remove chance for concurrency errors (e.g., nondeterminism from execution order, not just low-level data races)
- Efficiency Layer: Check for subtle concurrency bugs (races, deadlocks, and so on)
  - Mixture of verification and automated directed testing
  - Error detection on framework and libraries

21st Century Code Generation

- Problem: generating optimal code is like searching for needle in a haystack



- Manycore ⇒ even more diverse
- New approach: "Auto-tuners"
  - 1st generate program variations of combinations of optimizations (blocking, prefetching, …) and data structures
  - Then compile and run to heuristically search for best code for *that* computer
- Examples: PHiPAC (BLAS), Atlas (BLAS), Spiral (DSP), FFT-W (FFT)
- Example: Sparse Matrix (SpMV) for 4 multicores
  - Fastest SpMV: 2X OSKI/PETSc Intel Clovertown, 4X AMD Opteron
  - Optimization space: register blocking, cache blocking, TLB blocking, prefetching/DMA options, NUMA, BCOO v. BCSR data structures, 16b v. 32b indices, loop unrolling, …

Example: Sparse Matrix * Vector

Name	Clovertwn	Opteron	Cell	Niagara 2
Chips*Cores	2*4 = 8	2*2 = 4	1*8 = 8	1*8 = 8
Architecture	4-/3-issue, SSE3, OOO, caches		2-VLIW, SIMD,RAM	1-issue, MT,cache
Clock Rate	2.3 GHz	2.2 GHz	3.2 GHz	1.4 GHz
Peak MemBW	21 GB/s	21 GB/s	26 GB/s	41 GB/s
Peak GFLOPS	74.6 GF	17.6 GF	14.6 GF	11.2 GF

20th Century Metrics: Clock Rate or Theoretical Peak Performance

Example: Sparse Matrix * Vector

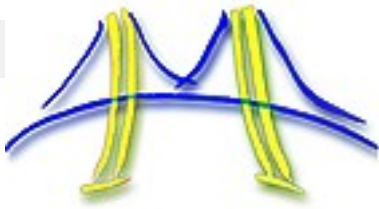
Name	Clovertwn	Opteron	Cell	Niagara 2
Chips*Cores	2*4 = 8	2*2 = 4	1*8 = 8	1*8 = 8
Architecture	4-/3-issue, SSE3, OOO, caches, prefch		2-VLIW, SIMD,RAM	1-issue, cache,MT
Clock Rate	2.3 GHz	2.2 GHz	3.2 GHz	1.4 GHz
Peak MemBW	21 GB/s	21 GB/s	26 GB/s	41 GB/s
Peak GFLOPS	74.6 GF	17.6 GF	14.6 GF	11.2 GF
Naïve SpMV (median of many matrices)	1.0 GF	0.6 GF	--	2.7 GF
Efficiency %	1%	3%	--	24%

21st Century: Actual (Autotuned) Performance

Example: Sparse Matrix * Vector

Name	Clovertwn	Opteron	Cell	Niagara 2
Chips*Cores	2*4 = 8	2*2 = 4	1*8 = 8	1*8 = 8
Architecture	4-/3-issue, SSE3, OOO, caches, prefch		2-VLIW, SIMD,RAM	1-issue, cache,MT
Clock Rate	2.3 GHz	2.2 GHz	3.2 GHz	1.4 GHz
Peak MemBW	21 GB/s	21 GB/s	26 GB/s	41 GB/s
Peak GFLOPS	74.6 GF	17.6 GF	14.6 GF	11.2 GF
Naïve SpMV (median of many matrices)	1.0 GF	0.6 GF	--	2.7 GF
Efficiency %	1%	3%	--	24%
Autotuned	1.5 GF	1.9 GF	**3.4 GF**	2.9 GF
Auto Speedup	**1.5X**	**3.2X**	∞	1.1X

Theme 3: Summary



- Greater productivity and efficiency for SpMV?

Parallelizing compiler +
Multicore +
Multilevel caches +
SW and HW prefetching

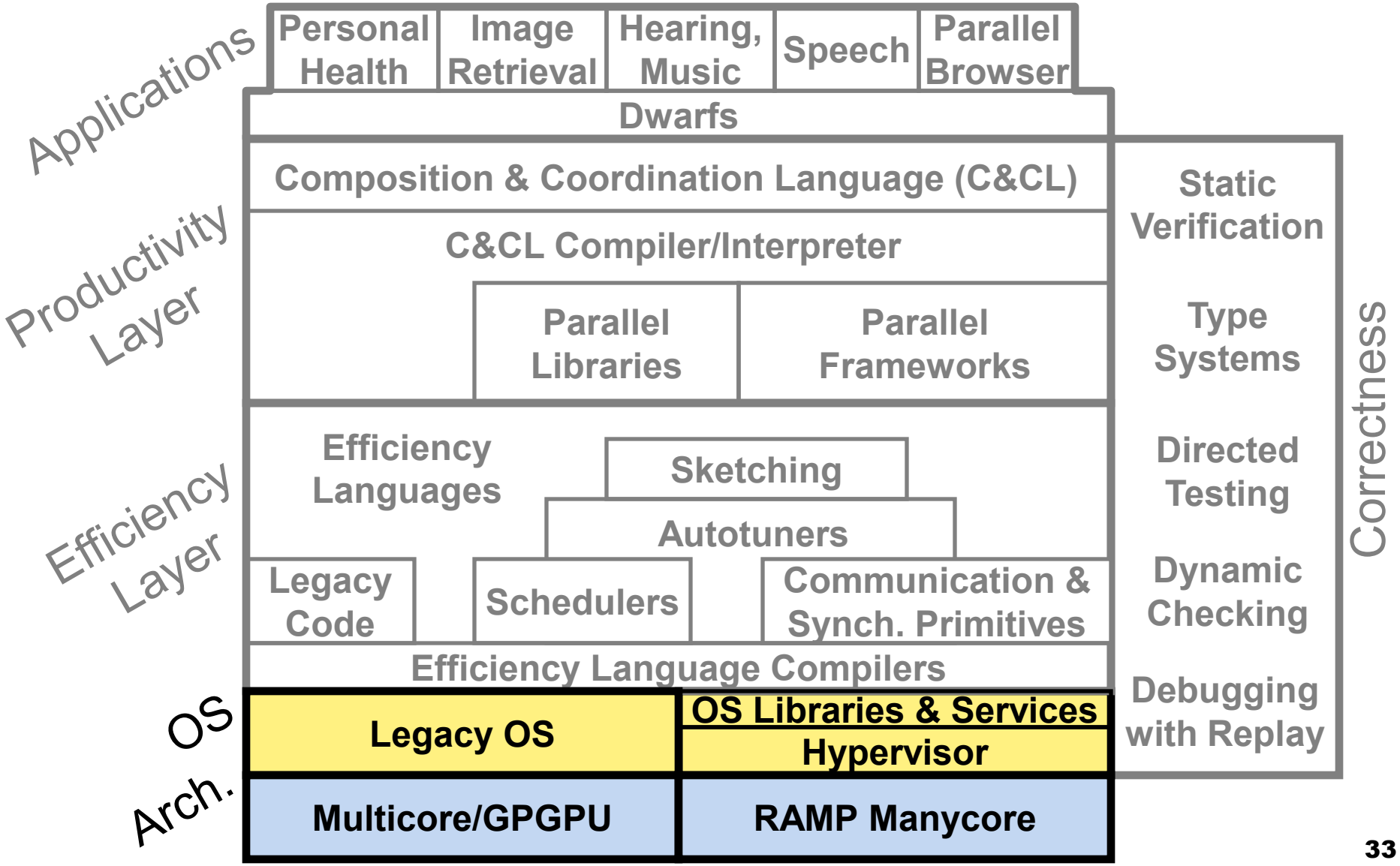
Dwarf focus +
Autotuner +
Multicore +
Local RAM +
DMA

SpMV: Easier to autotune single local RAM + DMA than multilevel caches + HW and SW prefetching

- Libraries and Frameworks to leverage experts
- Productivity Layer & Efficiency Layer
- C&C Language to compose Libraries/Frameworks

Par Lab Research Overview

Easy to write correct programs that run efficiently on manycore

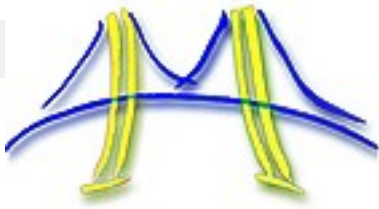


Theme 4: OS and Architecture

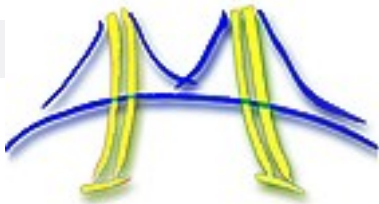
(Krste Asanovic, John Kubiatowicz)

- Traditional OSes brittle, insecure, memory hogs
  - Traditional monolithic OS image uses lots of precious memory * 100s - 1000s times (e.g., AIX uses GBs of DRAM / CPU)
- How can novel architectural support improve productivity, efficiency, and correctness for scalable hardware?
  - Efficiency instead of performance to capture energy as well as performance
- Other challenges: power limit, design and verification costs, low yield, higher error rates
- How prototype ideas fast enough to run real SW?

Deconstructing Operating Systems



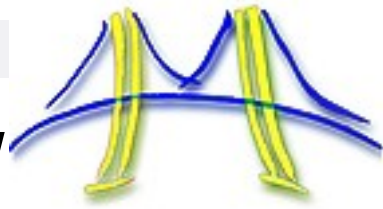
- Resurgence of interest in virtual machines
  - Hypervisor: thin SW layer btw guest OS and HW
- Future OS: libraries where only functions needed are linked into app, on top of thin hypervisor providing protection and sharing of resources
  - Opportunity for OS innovation
- Leverage HW partitioning support for very thin hypervisors, and to allow software full access to hardware within partition



HW Solution: Small is Beautiful

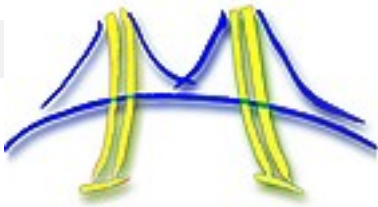
- Expect modestly pipelined (5- to 9-stage) CPUs, FPUs, vector, SIMD PEs
  - Small cores not much slower than large cores
- Parallel is energy efficient path to performance: CV^2F
  - Lower threshold and supply voltages lowers energy per op
- Redundant processors can improve chip yield
  - Cisco Metro 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 $\Rightarrow$ Heterogeneous processors
  - Special function units to accelerate popular functions

HW features supporting Parallel SW

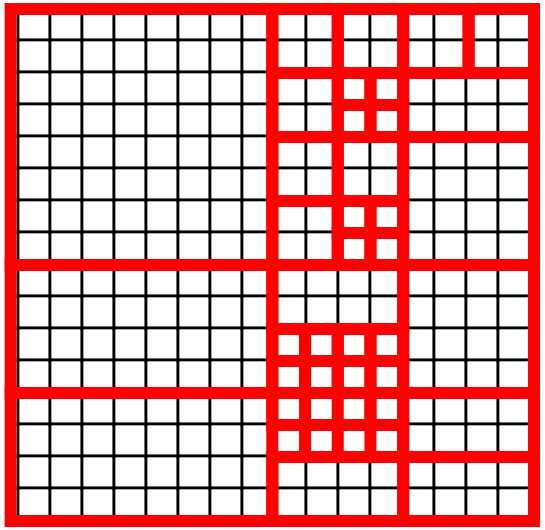


- Want Software Composable Primitives, Not Hardware Packaged Solutions
  - "You're not going fast if you're headed in the wrong direction"
  - Transactional Memory is usually a Packaged Solution
- Configurable Memory Hierarchy (Cell v. Clovertown)
  - Can configure on-chip memory as cache or local RAM
  - Programmable DMA to move data without occupying CPU
  - Cache coherence: Mostly HW but SW handlers for complex cases
  - Hardware logging of memory writes to allow rollback
- Active messages plus user-level event handling
  - Used by parallel language runtimes to provide fast communication, synchronization, thread scheduling
- Partitions
- Fast Barrier Synchronization & Atomic Fetch-and-Op

Partitions and Fast Barrier Network



InfiniCore chip with 16x16 tile array



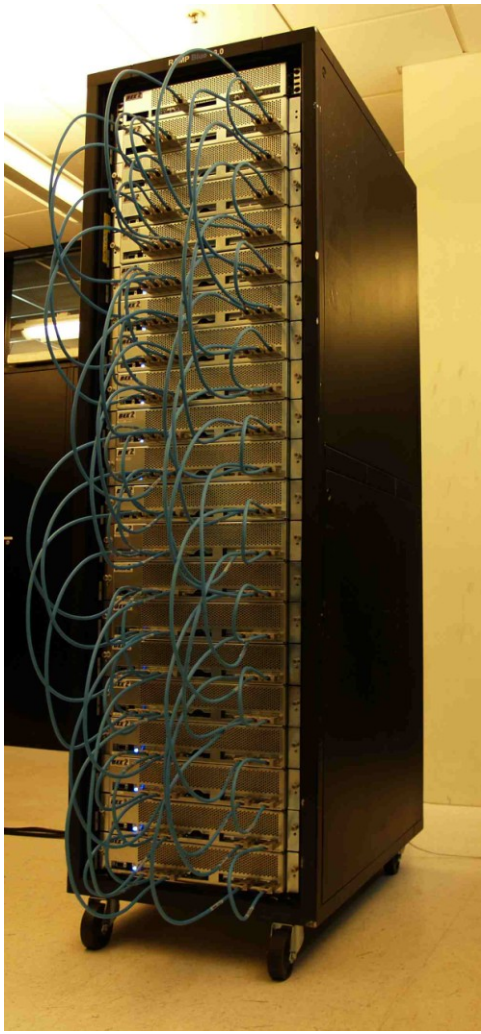
- Partition: hardware-isolated group
  - Chip divided into hardware-isolated partition, under control of supervisor software
  - User-level software has almost complete control of hardware inside partition
- Fast Barrier Network per partition (≈ 1ns)
  - Signals propagate combinationally
  - Hypervisor sets taps saying where partition sees barrier

Build Academic Manycore from FPGAs

- As ≈ 10 CPUs will fit in Field Programmable Gate Array (FPGA), 1000-CPU system from ≈ 100 FPGAs?
  - 8 32-bit simple "soft core" RISC at 100MHz in 2004 (Virtex-II)
  - FPGA generations every 1.5 yrs; ≈ 2X CPUs, ≈ 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
  - Ideal for heterogeneous chip architectures
  - RAMPants: 10 faculty at Berkeley, CMU, MIT, Stanford, Texas, and Washington
- "Research Accelerator for Multiple Processors" as a vehicle to lure more researchers to parallel challenge and decrease time to parallel solution

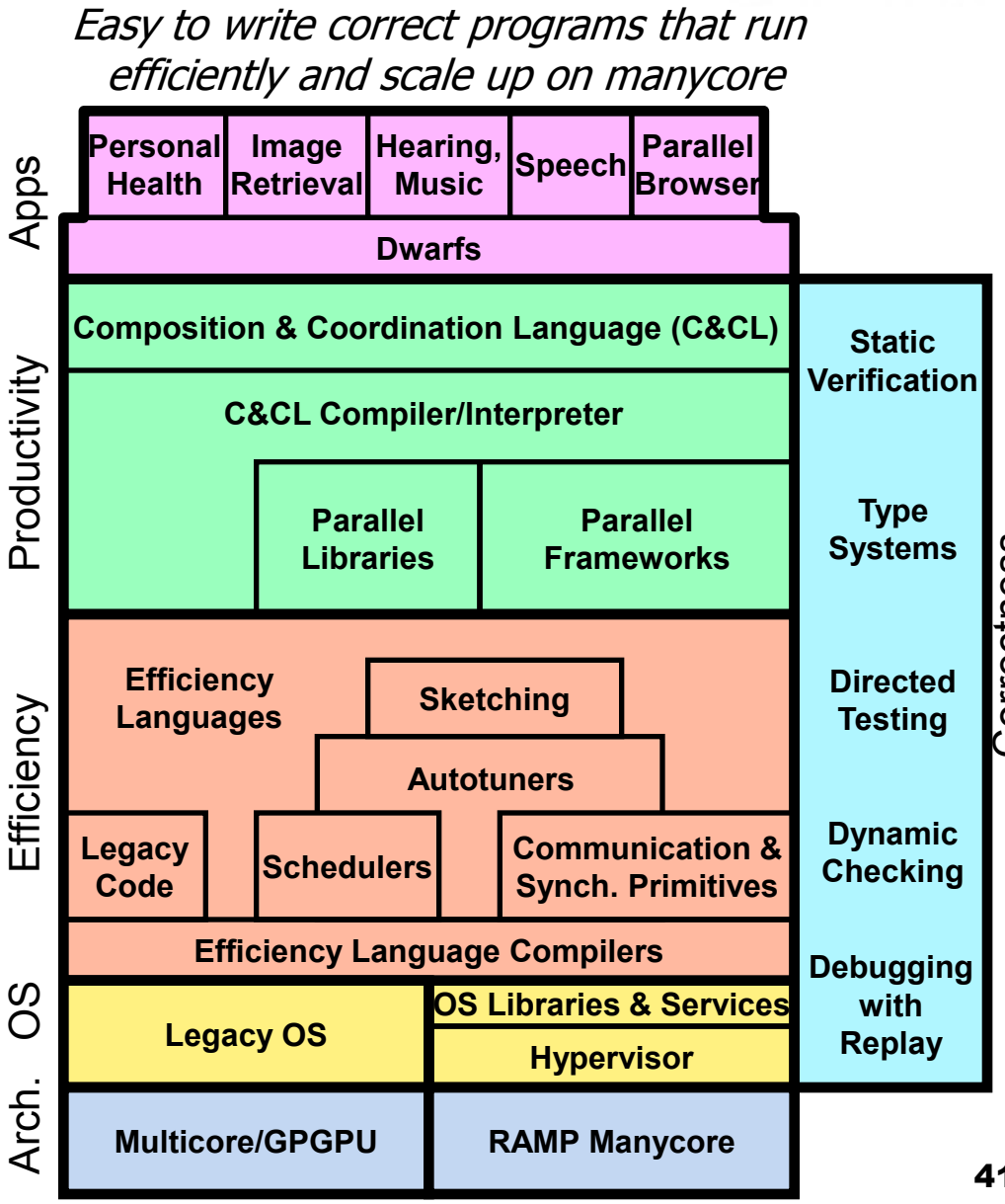
1008 Core "RAMP Blue" (Wawrzynek, Krasnov,… at Berkeley)

- 1008 = 12 32-bit RISC cores / FPGA, 4 FGPAs/board, 21 boards
  - Simple MicroBlaze soft cores @ 90 MHz
    - Full star-connection between modules
- NASA Advanced Supercomputing (NAS) Parallel Benchmarks (all class S)
  - UPC versions (C plus shared-memory abstraction) CG, EP, IS, MG
- RAMPants creating HW & SW for many-core community using next gen FPGAs
  - Chuck Thacker & Microsoft designing next boards
  - 3rd party to manufacture and sell boards: 1H08
  - Gateware, Software BSD open source



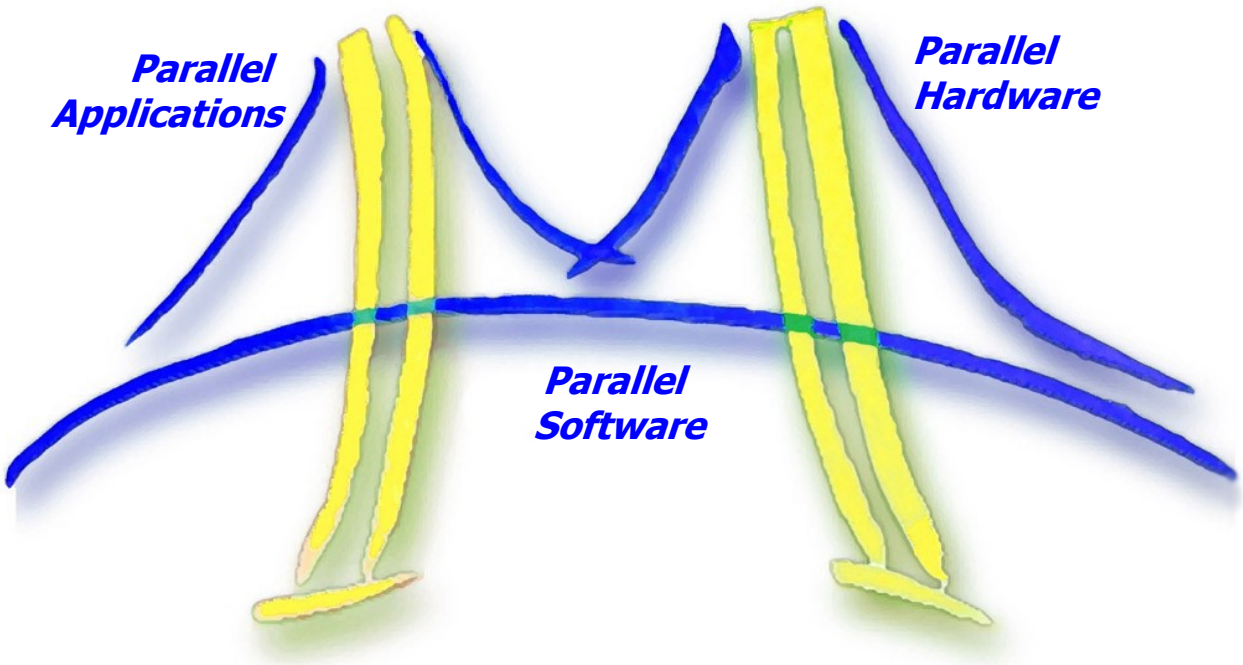
Summary: A Berkeley View 2.0

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



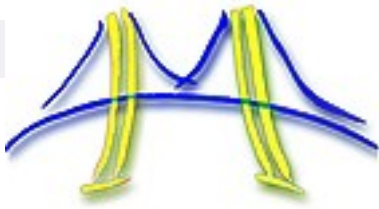
Acknowledgments

- Berkeley View material comes from discussions with:
  - Profs Krste Asanovic, Ras Bodik, Jim Demmel, Kurt Keutzer, John Kubiatowicz, John Wawrzynek, Kathy Yelick
  - UCB Grad students Bryan Catanzaro, Jike Chong, Joe Gebis, William Plishker, and Sam Williams
  - LBNL:Parry Husbands, Bill Kramer, Lenny Oliker, John Shalf
- **See view.eecs.berkeley.edu**
- RAMP based on work of RAMP Developers:
  - Krste Asanovic (Berkeley), Derek Chiou (Texas), James Hoe (CMU), Christos Kozyrakis (Stanford), Shih-Lien Lu (Intel), Mark Oskin (Washington), David Patterson (Berkeley, Co-PI), and John Wawrzynek (Berkeley, PI)
- **See ramp.eecs.berkeley.edu**



Backup Slides

RAD Lab 5-year Mission

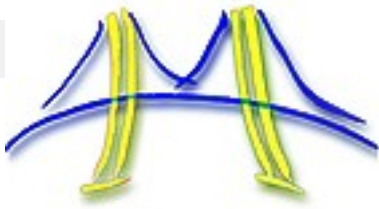


- Today's Internet systems complex, fragile, manually managed, rapidly evolving
  - To scale Ebay, must build Ebay-sized company
- "Moon shot" mission statement:

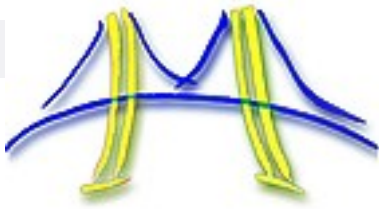
Enable *a single person* to Develop, Assess, Deploy, and Operate the next-generation IT service

  - "The Fortune 1 Million" by enabling rapid innovation
- Create core technology to enable vision via synergy across systems, networking, and Statistical Machine Learning
- **Making datacenter easier to manage enables vision of single person to analyze, deploy and operate a scalable IT service**

If Datacenter is the computer…



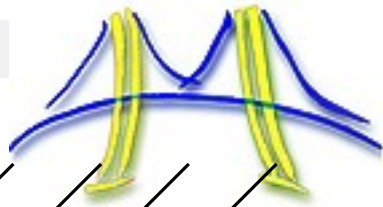
- What is the programming language?
  - Ruby on Rails - Group CS 198 / 98
- What are the libraries?
- How do trace/monitor programs?
- What is the simulator?
- What is Computer Aided Design?
- What is the Operating System?
- What is the Database System?



How to succeed at the hardest problem to face computer science?

- Try different approaches from the past
- Recruit the best minds to help
- Academic & industrial research
  - Led to 19 multibillion dollar IT industries

Universities help start 19 1B$+ Industries

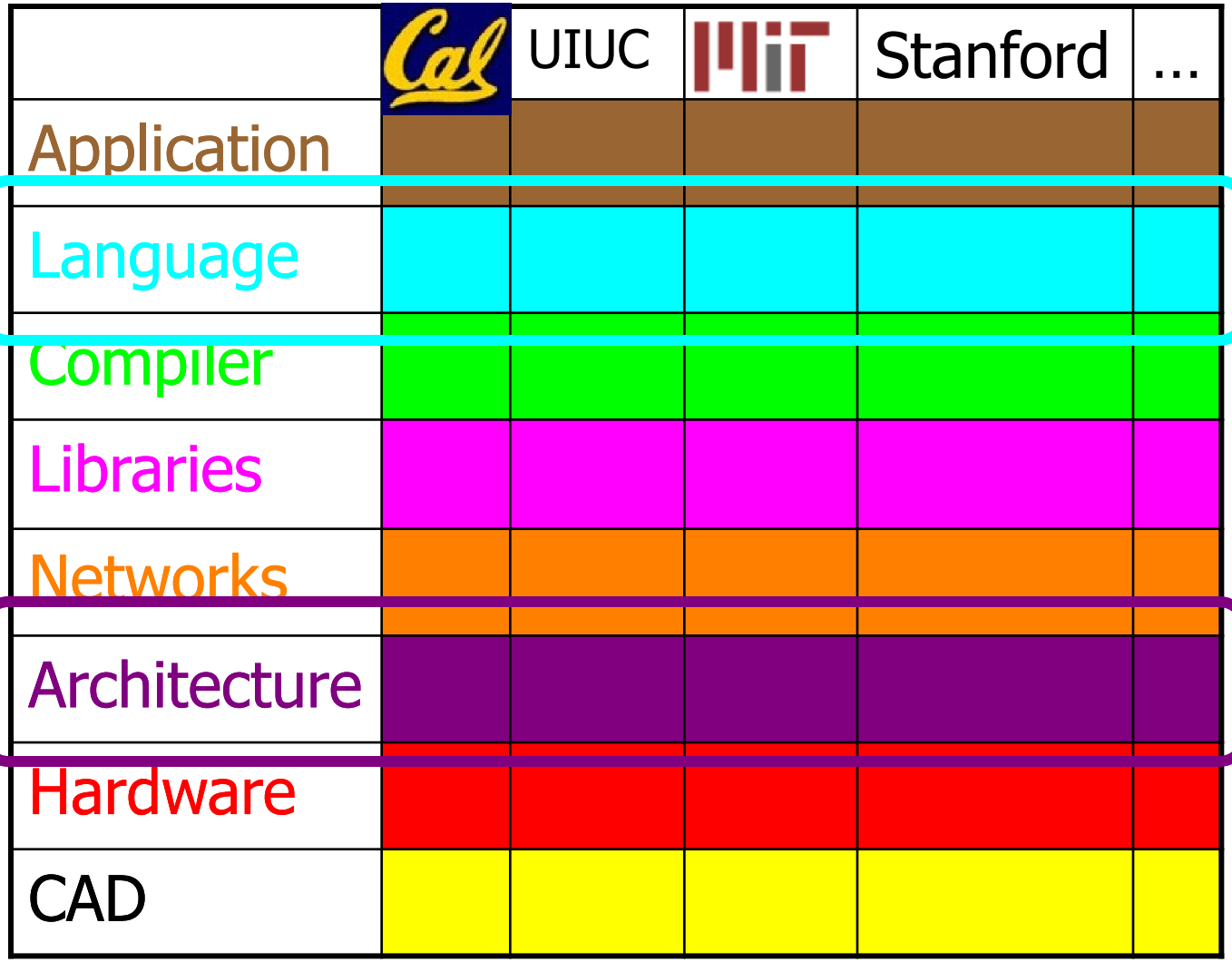


	$1B+ Industry	Berkeley	Caltech	CERN	CMU	Illinois	MIT	Purdue	Rochest.	Stanford	Tokyo	UCLA	Utah	Wisc.
1	Timesharing													
2	Client/server													
3	Graphics													
4	Entertainment													
5	Internet													
6	LANs													
7	Workstations													
8	GUI													
9	VLSI design													
10	RISC processors													
11	Relational DB													
12	Parallel DB													
13	Data mining													
14	Parallel computing													
15	RAID disk arrays													
16	Portable comm.													
17	World Wide Web													
18	Speech recognition													
19	Broadband last mile													
	Total	7	2	2	3	2	5	1	2	4	1	3	1	3

Source: *Innovation in Information Technology*, National Research Council Press, 2003.

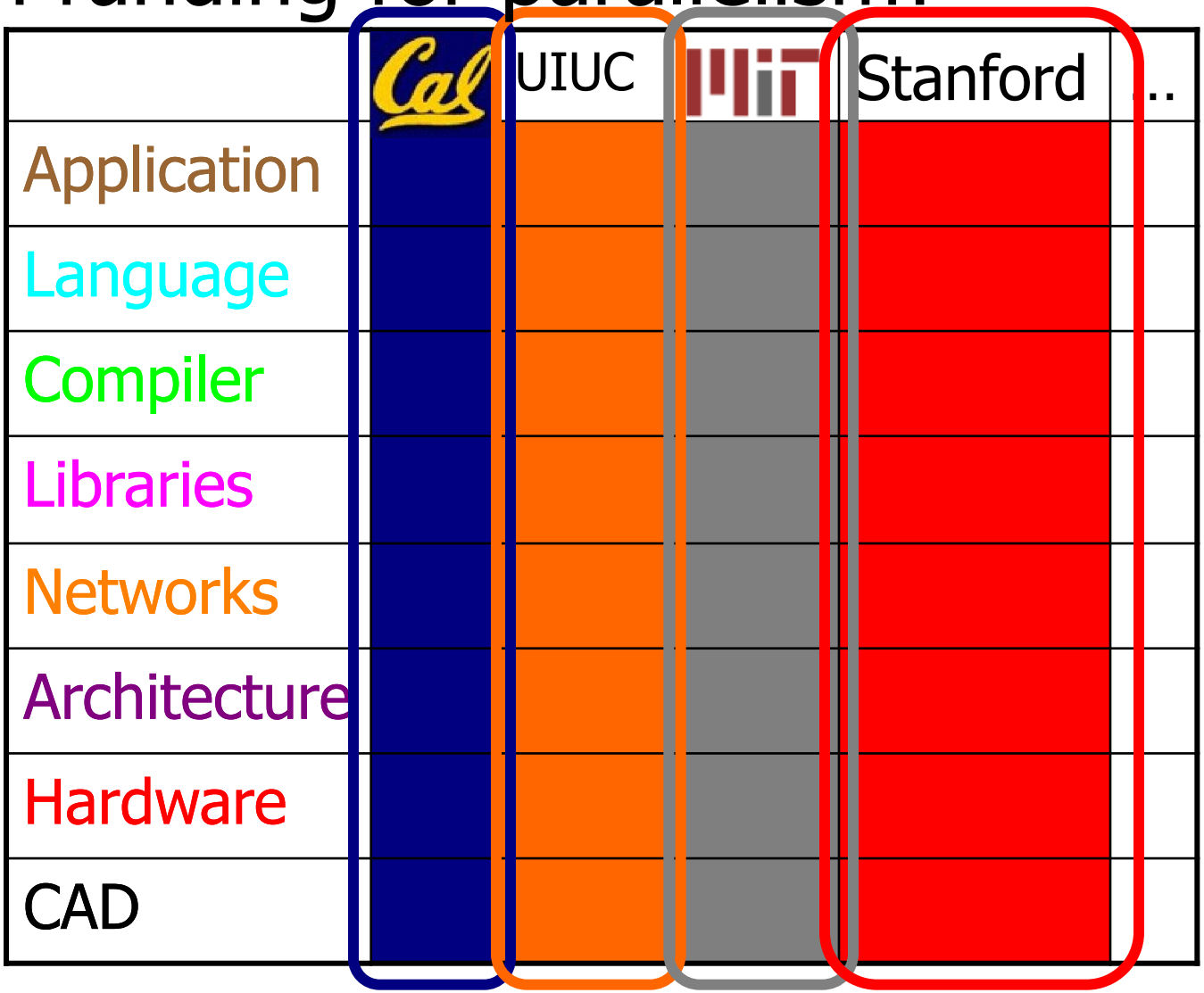
Change directions of research funding for parallelism?

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

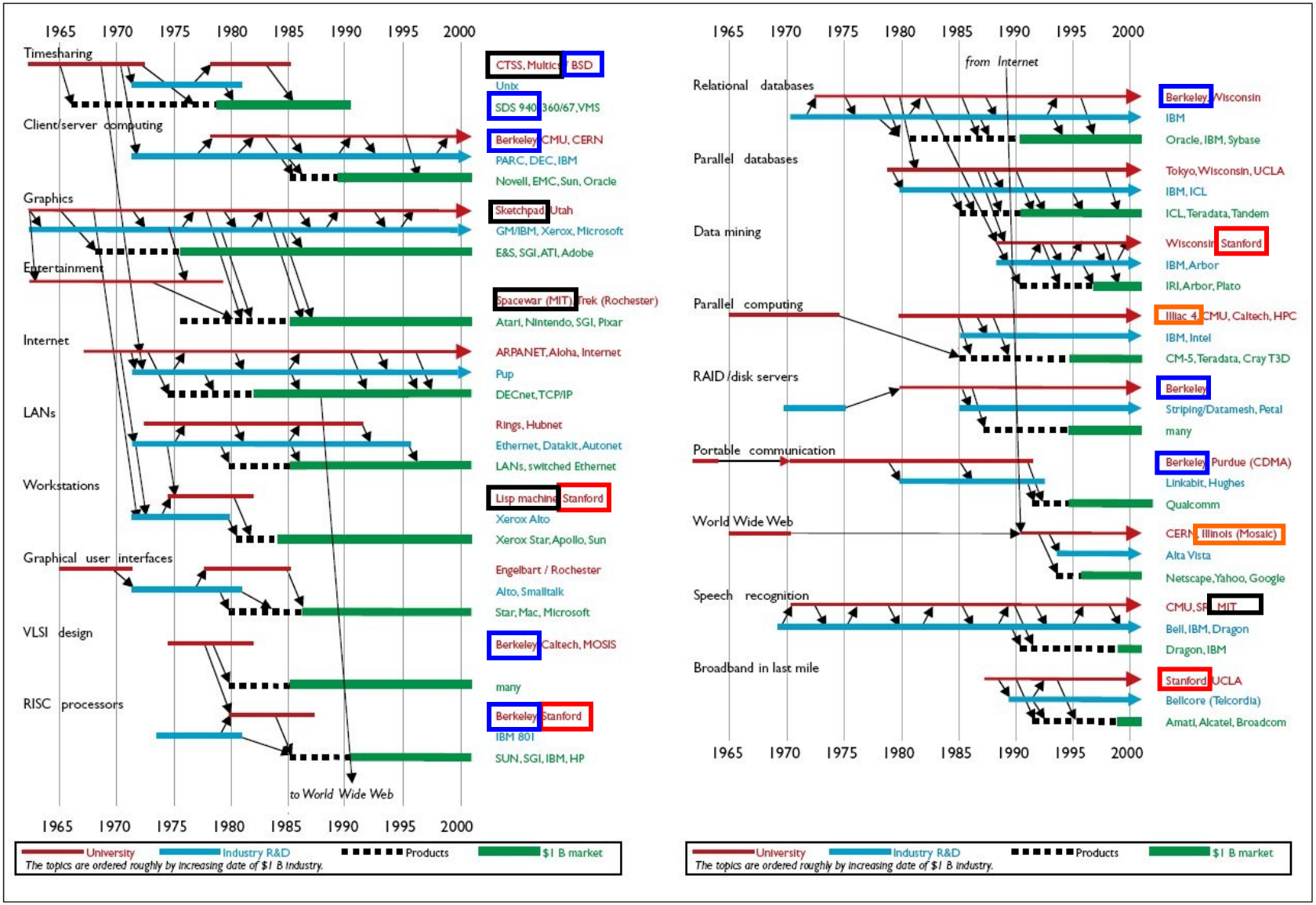


Change directions of research funding for parallelism?

To increase cross-disciplinary bandwidth, get experts **per site** collabor-ating on (app-driven) parallelism



Research ⇒ $1B+ Industries



Source: From [6], reprinted with permission from the National Academy of Sciences, courtesy of the National Academies Press, Washington D.C. ©2003.

Universities help start 19 1B$+ Industries

$1B+ Industry	Berkeley	Caltech	CERN	CMU	Illinois	MIT	Purdue	Rochest.	Stanford	Tokyo	UCLA	Utah	Wisc.
9 VLSI design	■	■											
10 RISC processors	■								■				
11 Relational DB	■												■
12 Parallel DB										■			■
13 Data mining									■				
14 Parallel computing		■		■	■								
15 RAID disk arrays	■												
16 Portable comm.	■		■				■						
17 World Wide Web					■								
18 Speech recognition				■		■							
19 Broadband last mile									■		■		
Total	5	2	1	2	2	1	1	1	3	1	1	0	2

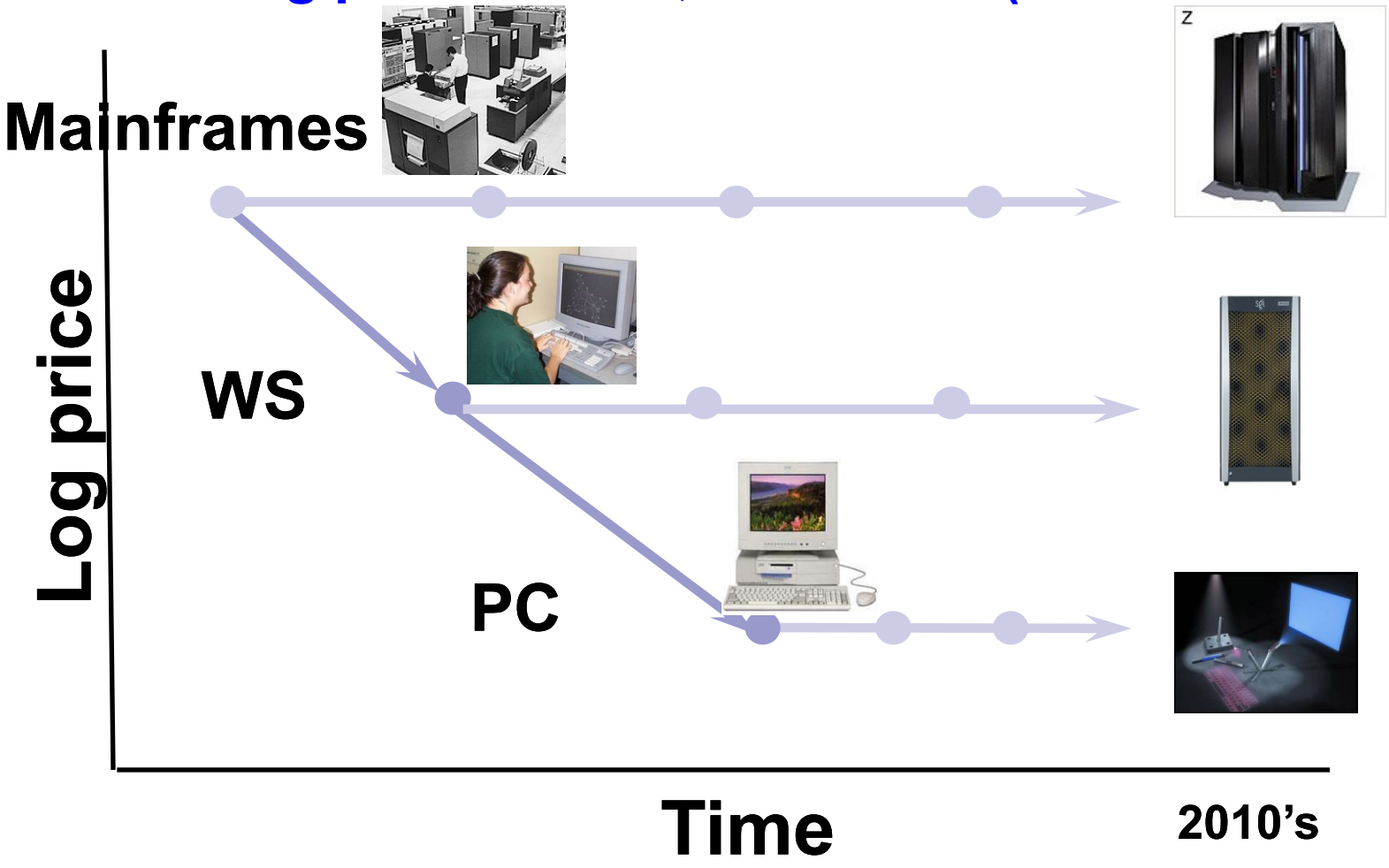
Last ≈ 25 years

Source: *Innovation in Information Technology*, National Research Council Press, 2003.

Bell's Evolution Of Computer Classes

Technology enables two paths:

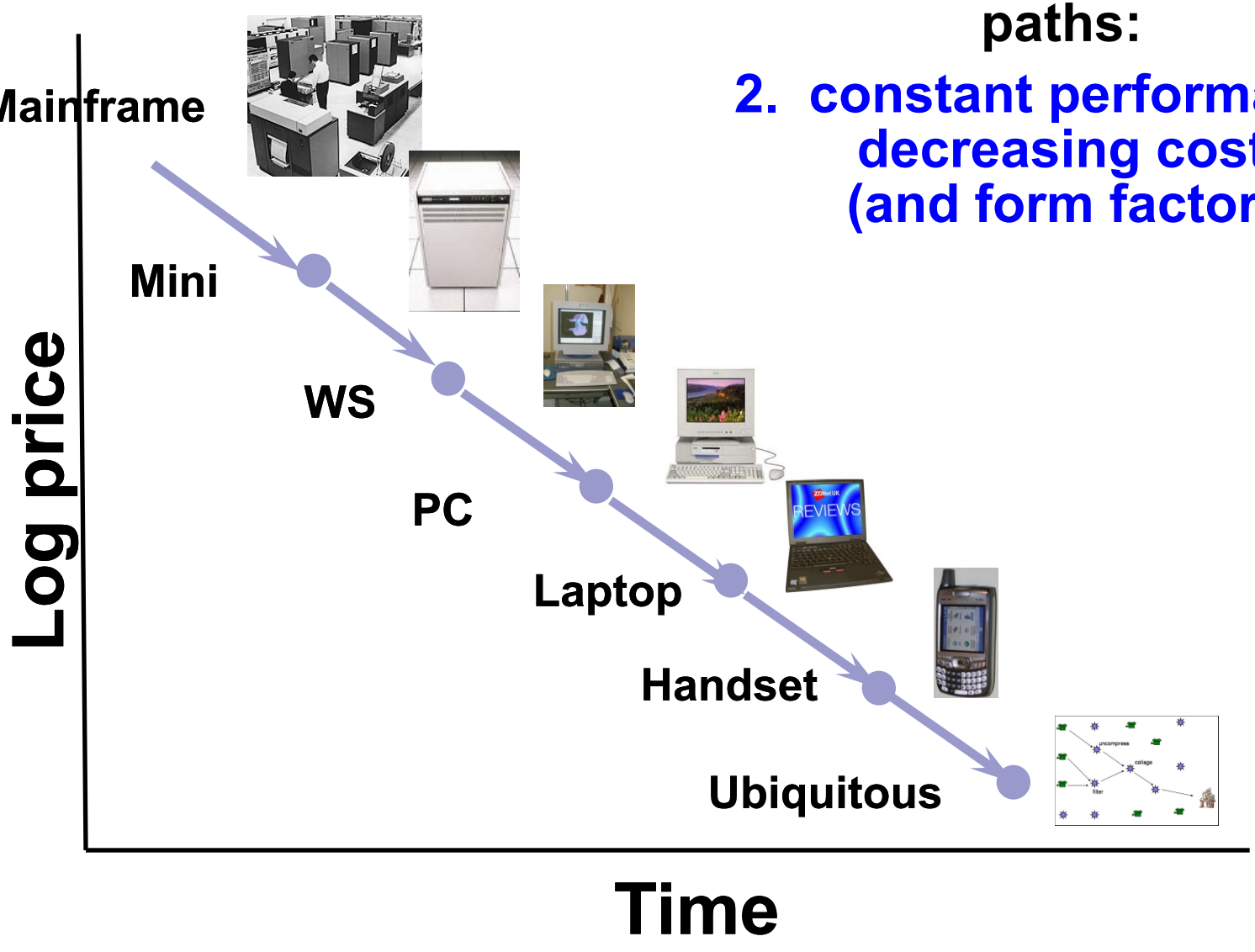
1. **increasing performance, same cost (and form factor)**



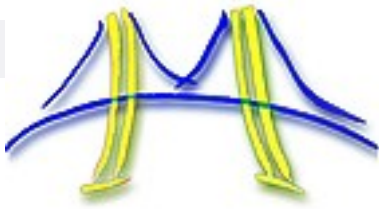
Bell's Evolution Of Computer Classes

Technology enables two paths:

2. constant performance, decreasing cost (and form factor)

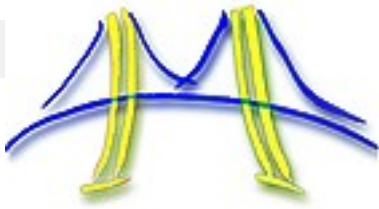


Example: Sparse Matrix * Vector



Name	Clovertown	Opteron	Cell
Chips*Cores	2*4 = 8	2*2 = 4	1*8 = 8
Architecture	4-/3-issue, 2-/1-SSE3, OOO, caches, prefetch		2-VLIW, SIMD, local RAM, DMA
Clock Rate	2.3 GHz	2.2 GHz	3.2 GHz
Peak MemBW	21.3 GB/s	21.3	25.6 GB/s
SpMV MemBW	7.5 GB/s	10.0	22.5 GB/s
Efficiency %	35%	47%	88%
Peak GFLOPS	74.6	17.6	14.6 (DP Fl. Pt.)
SpMV GFLOPS	1.5	1.9	3.4
Efficiency %	2%	11%	23%

Number of Cores/Chip: Manycore



- We need revolution, not evolution
- "Multicore" 2X cores per generation: 2, 4, 8, …
- "Manycore" 100s is highest performance per unit area, and per Watt, then 2X per generation: 64, 128, 256, 512, 1024 …
- Multicore architectures & Programming Models good for 2 to 32 cores won't evolve to Manycore systems of 100's of processors
  ⇒ Desperately need HW/SW models that work for Manycore or will run out of steam
  (as ILP ran out of steam at 4 instructions)

Par Lab Organization

- Follow ACM Special Interest Groups/IEEE TC
  - Allow Par Lab researchers to join multiple SIG meetings
  - SIG "Chairs" meet monthly dinner meeting (SIG Board)