

The Parallel Revolution Has Started: Are You Part of the Solution or Part of the Problem?

Dave Patterson
Parallel Computing Laboratory
U.C. Berkeley
June, 2008



Outline

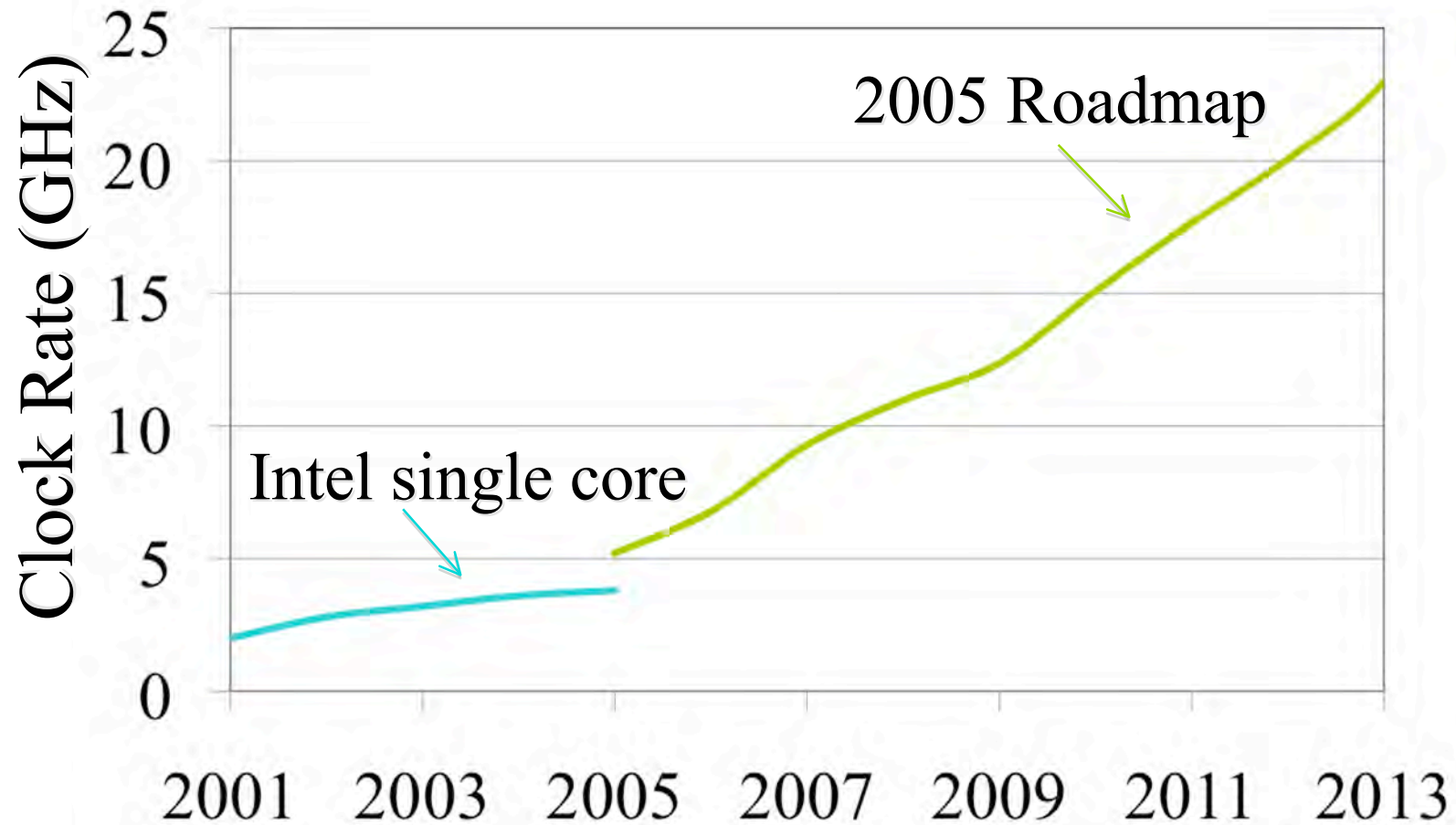
- What Caused the Revolution?
- Is it Too Late to Stop It?
- Is it an Interesting, Important Research Problem or Just Doing Industry's Dirty Work?
- Why Might We Succeed (this time)?
- Projected Hardware/Software Context?
- Example Coordinated Attack: Par Lab @ UCB
- Conclusion



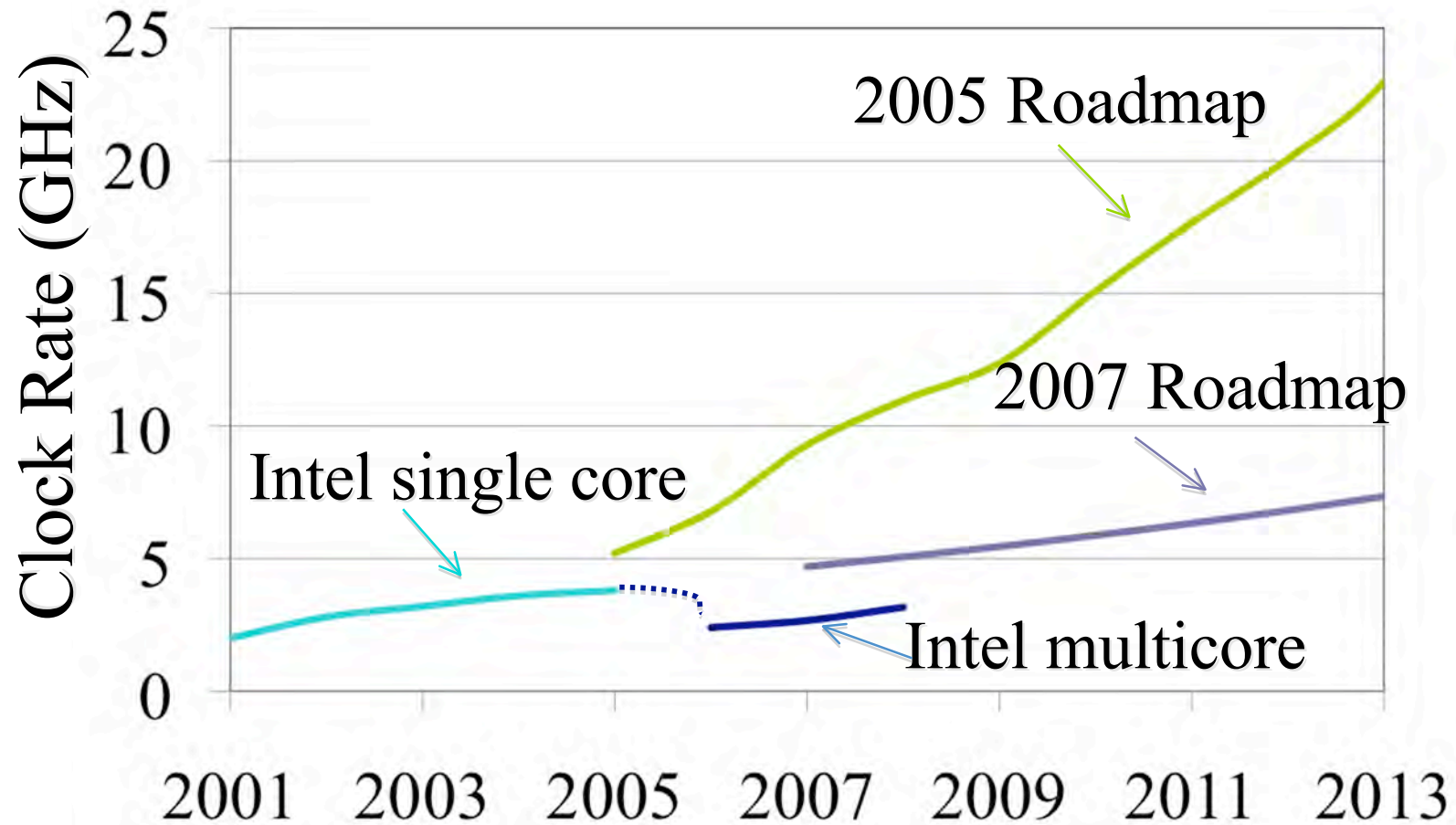
A Parallel Revolution, Ready or Not

- 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 \approx 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

2005 IT Roadmap Semiconductors



Change in ITS Roadmap in 2 yrs



You can't prevent the start of the revolution

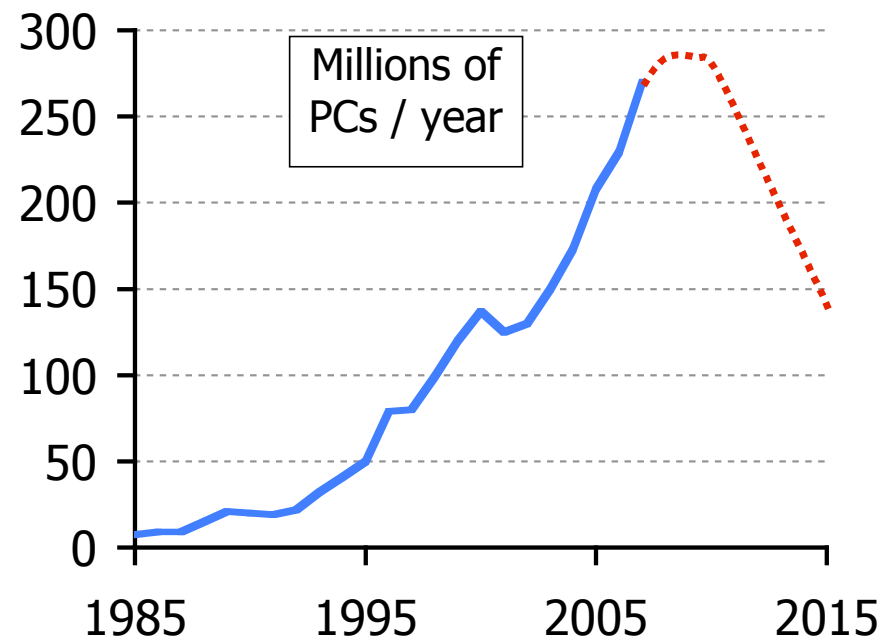


- While evolution and global warming are “controversial” in scientific circles, belief in need to switch to parallel computing is unanimous in the hardware community
- AMD, Intel, IBM, Sun, ... now sell more multiprocessor (“multicore”) chips than uniprocessor chips
 - Plan on little improvement in clock rate (8% / year?)
 - Expect 2X cores every 2 years, ready or not
 - Note – they are already designing the chips that will appear over the next 5 years, and they’re parallel

But Parallel Revolution May Fail



- 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
 - ⇒ Fewer jobs in IT industry



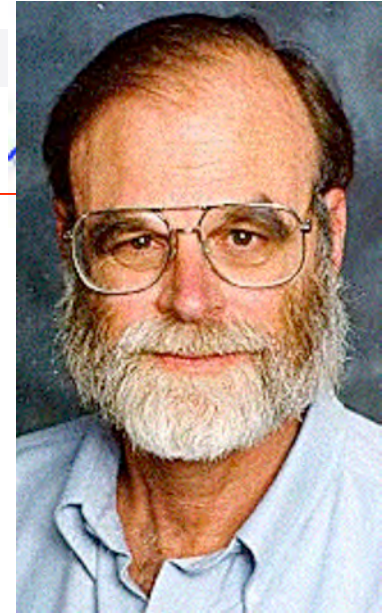


How important and difficult is parallel computing research?



- Jim Gray's 12 Grand Challenges as part of Turing Award Lecture in 1998
 - Examined all past Turing Award Lectures
 - Develop list for 21st Century
- Gartner 7 IT Grand Challenges in 2008
 - a fundamental issue to be overcome within the field of IT whose resolutions will have broad and extremely beneficial economic, scientific or societal effects on all aspects of our lives.
- David Kirk, NVIDIA, 10 Challenges in 2008
- John Hennessy's assessment of parallelism

Gray's List of 12 Grand Challenges



1. Devise an architecture that scales up by 10^6 .
2. The Turing test: win the impersonation game 30% of time.
 - a. 3. Read and understand as well as a human.
 - b. 4. Think and write as well as a human.
3. Hear as well as a person (native speaker): speech to text.
4. Speak as well as a person (native speaker): text to speech.
5. See as well as a person (recognize).
6. Remember what is seen and heard and quickly return it on request.
7. Build a system that, given a text corpus, can answer questions about the text and summarize it as quickly and precisely as a human expert. Then add sounds: conversations, music. Then add images, pictures, art, movies.
8. Simulate being some other place as an observer (Tele-Past) and a participant (Tele-Present).
9. Build a system used by millions of people each day but administered by a _ time person.
10. Do 9 and prove it only services authorized users.
11. Do 9 and prove it is almost always available: (out 1 sec. per 100 years).
12. Automatic Programming: Given a specification, build a system that implements the spec. Prove that the implementation matches the spec. Do it better than a team of programmers.



Gartner 7 IT Grand Challenges



1. Never having to manually recharge devices
2. **Parallel Programming**
3. Non Tactile, Natural Computing Interface
4. Automated Speech Translation
5. Persistent and Reliable Long-Term Storage
6. Increase Programmer Productivity 100-fold
7. Identifying the Financial Consequences of IT Investing

David Kirk's (NVIDIA) Top 10



1. Reliable Software
2. Reliable Hardware
3. **Parallel Programming**
4. Memory Power, Size, Bandwidth Walls
5. Locality:
Eliminate/Respect
Space-time constraints
6. Threading: MIMD, SIMD, SIMT
7. Secure Computing
8. Compelling U.I.
9. Extensible Distrib. Computing
10. Interconnect
11. Power

Keynote Address, 6/24/08, Int'l Symposium on Computer Architecture, Beijing, China



John Hennessy



- Computing Legend and President of Stanford University:

"...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."

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



Outline

- What Caused the Revolution?
- Is it Too Late to Stop It?
- Is it an Interesting, Important Research Problem or Just Doing Industry's Dirty Work?
- Why Might We Succeed (this time)?
- Projected Hardware/Software Context?
- Example Coordinated Attack: Par Lab @ UCB
- Conclusion



Why might we succeed this time?



■ No Killer Microprocessor

- No one is building a faster serial microprocessor
- Programmers needing more performance have no other option than parallel hardware

■ Vitality of Open Source Software

- OSS community is a meritocracy, so it's more likely to embrace technical advances
- OSS more significant commercially than in past

■ All the Wood Behind One Arrow

- Whole industry committed, so more people working on it



Why might we succeed this time?



- **Single-Chip Multiprocessors Enable Innovation**
 - Enables inventions that were impractical or uneconomical
- **FPGA prototypes shorten HW/SW cycle**
 - Fast enough to run whole SW stack, can change every day vs. every 5 years
- **Necessity Bolsters Courage**
 - Since we must find a solution, industry is more likely to take risks in trying potential solutions
- **Multicore Synergy with Software as a Service**

Context: 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





Context: Trends over Next Decade



- Flash memory replacing mechanical disks
 - Especially in portable client, but also increasing used along side disks in servers
- Expanding Software As A Service
 - Applications for the datacenter
 - Web 2.0 apps delivered via browser
 - Continue transition from shrink wrap software to services over the Internet
- Expanding “Hardware As A Service” (aka Cloud Computing aka Utility Computing)
 - New trend to outsource datacenter hardware
 - E.g, Amazon EC2/S3, Google Apps Engine, ...



Context: Excitement of Utility/Cloud Computing/HW as a Service

- 0\$ Capital for your own “Data Centers”
 - Pay as you go: for startups “S3 means no VC”
- Cost Associativity for Data Center: cost of 1000 servers x 1 hr = 1 server x 1000 hrs
- Data Center Price Model → Reward Conservation, “Just In Time” Provisioning
 - “Fast” scale-down → No dead or idle CPUs
 - “Instant” scale-up → No provisioning



Outline

- What Caused the Revolution?
- Is it Too Late to Stop It?
- Is it an Interesting, Important Research Problem or Just Doing Industry's Dirty Work?
- Why Might We Succeed (this time)?
- Projected Hardware/Software Context?
- **Example Coordinated Attack: Par Lab @ UCB**
- **Conclusion**



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 Kubiawicz, 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 12/2006 and new Parallel Computing Laboratory ("Par Lab")
- Goal: Productive, Efficient, Correct, Portable SW for 100+ cores & scale as double cores every 2 years (!)

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



5 Themes of Par Lab

1. Applications

Compelling apps drive top-down research agenda

2. Identify Common Design Patterns

Breaking through disciplinary boundaries

3. Developing Parallel Software with Productivity, Efficiency, and Correctness

2 Layers + Coordination & Composition Language
+ Autotuning

4. OS and Architecture

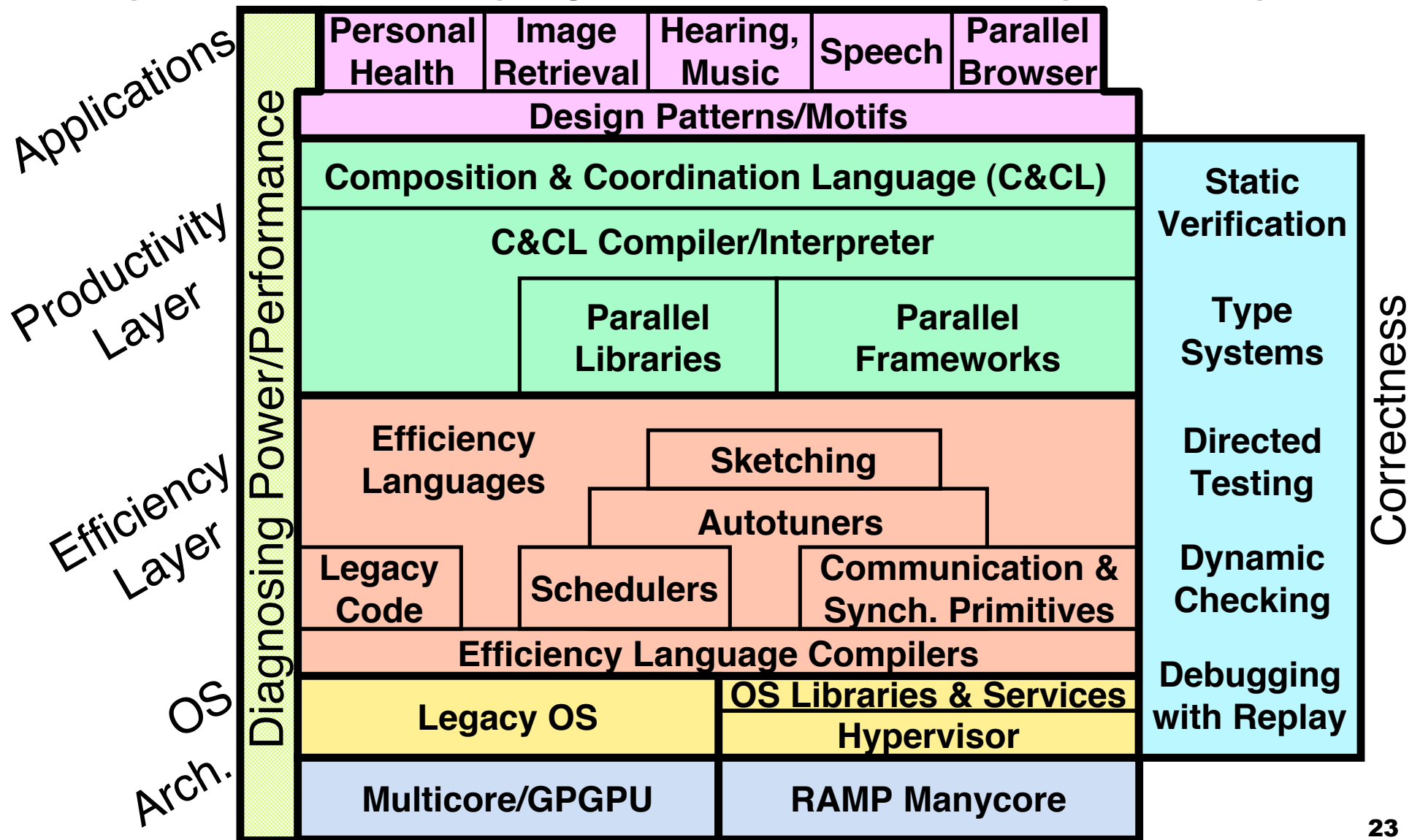
Composable primitives, not packaged solutions
Deconstruction, Fast barrier synchronization, Partitions

5. Diagnosing Power/Performance Bottlenecks

Par Lab Research Overview



Easy to write correct programs that run efficiently on manycore



Theme 1. Applications. What are the problems?

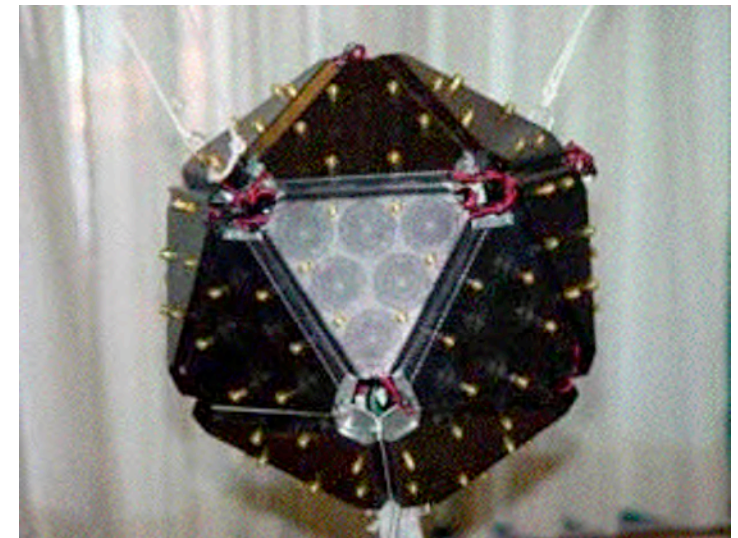


- “Who needs 100 cores to run M/S Word?”
 - Need compelling apps that use 100s of cores
- How did we pick applications?
 1. Enthusiastic expert application partner, leader in field, promise to help design, use, evaluate our technology
 2. Compelling in terms of likely market or social impact, with short 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)
 - Markets (consumer, business, health)



Compelling Laptop/Handheld Apps (David Wessel)

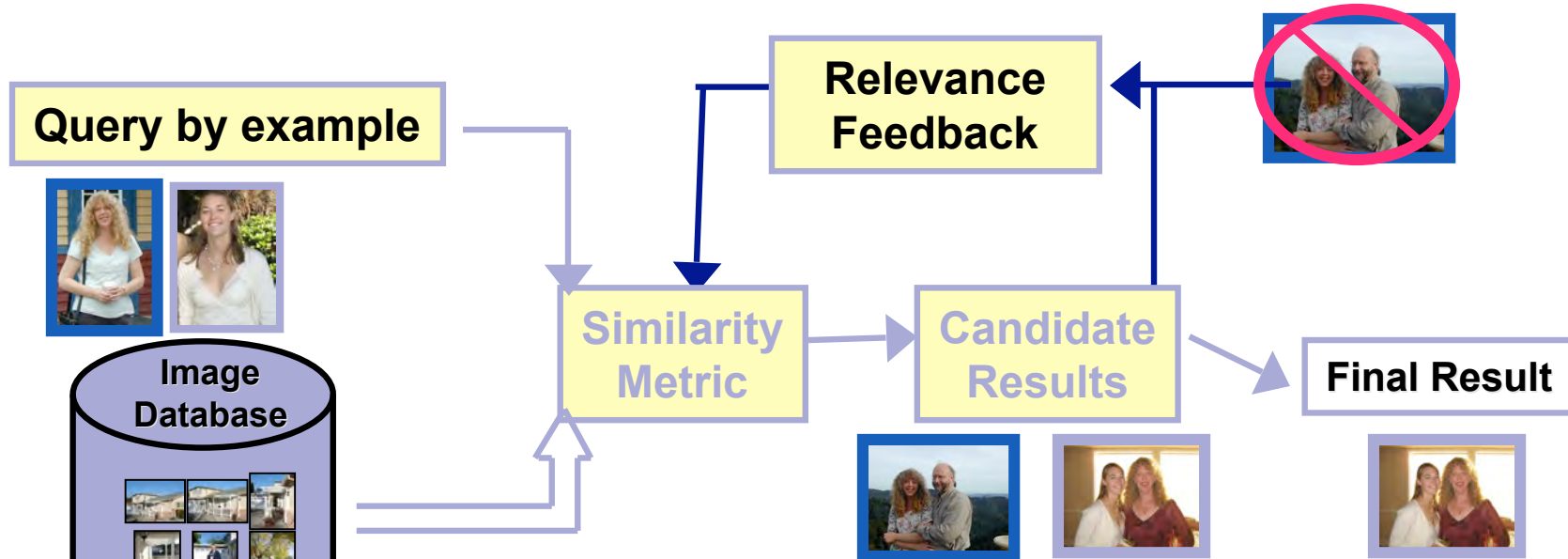
- Musicians have an insatiable appetite for computation
 - More channels, instruments, more processing, more interaction!
 - Latency must be low (5 ms)
 - Must be reliable (No clicks)
- 1. Music Enhancer
 - Enhanced sound delivery systems for home sound systems using large microphone and speaker arrays
 - 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.

Content-Based Image Retrieval

(Kurt Keutzer)

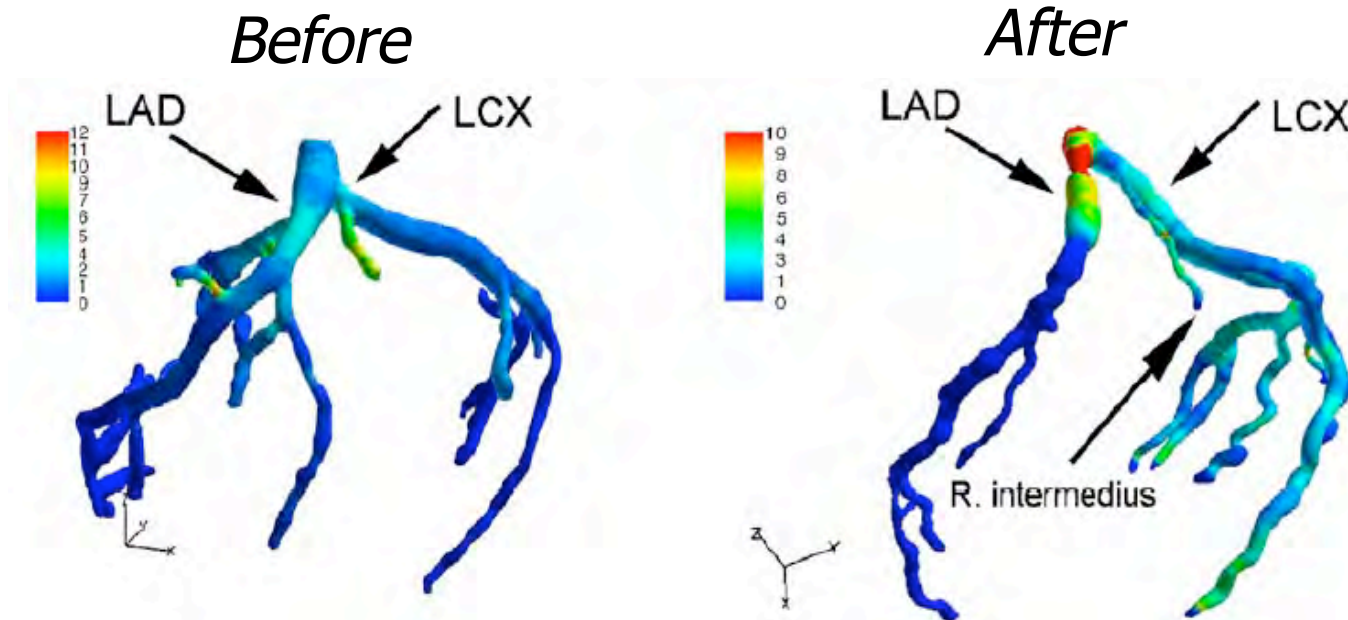


1000's of images



- 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

Coronary Artery Disease (Tony Keaveny)



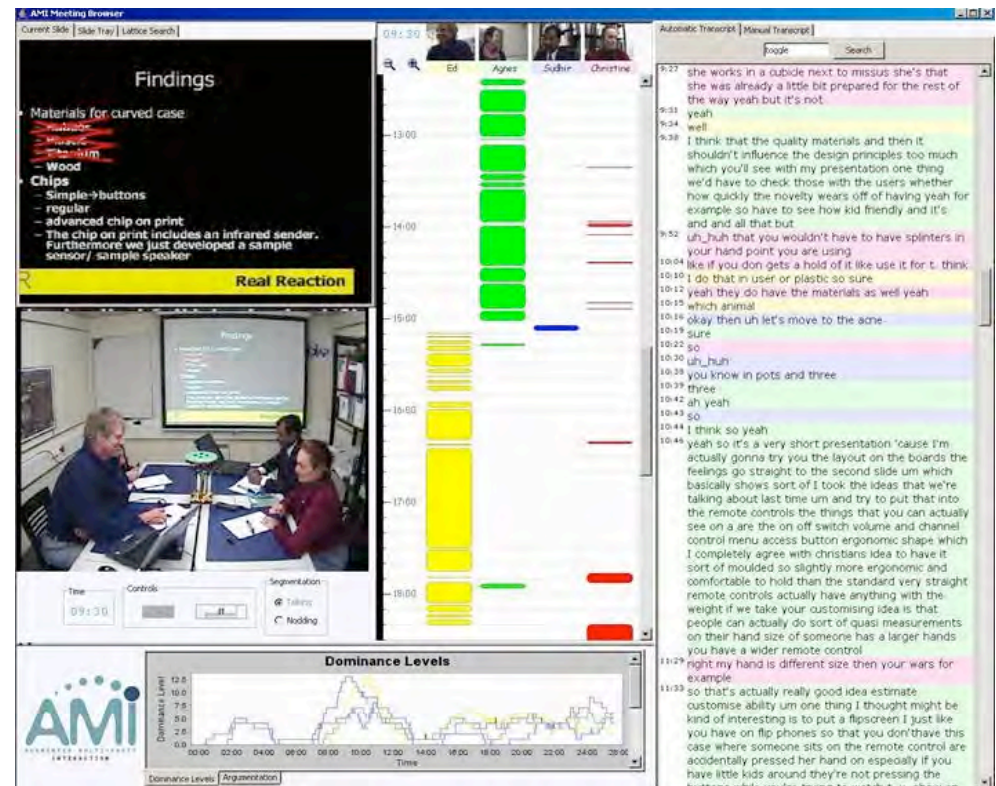
- Modeling to help patient compliance?
 - 450k deaths/year, 16M w. symptom, 72M ↑ BP
- Massively parallel, Real-time variations
 - CFD FE solid (non-linear), fluid (Newtonian), pulsatile
 - Blood pressure, activity, habitus, cholesterol

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

Parallel Browser (Ras Bodik)



- Web 2.0: Browser plays role of traditional OS
 - Resource sharing and allocation, Protection
- 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

Compelling Laptop/Handheld Apps



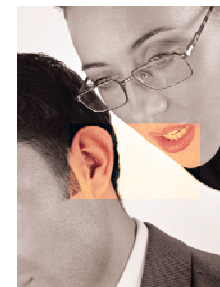
■ Health Coach

- 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)



Theme 2. Use design patterns



- How invent parallel systems of future when tied to old code, programming models, CPUs of the past?
- Look for common design patterns (see *A Pattern Language*, Christopher Alexander, 1975)
- *design patterns*: time-tested solutions to recurring problems in a well-defined context
 - “family of entrances” pattern to simplify comprehension of multiple entrances for a 1st-time visitor to a site
- *pattern “language”*: collection of related and interlocking patterns that flow into each other as the designer solves a design problem



Theme 2. What to compute?

- Look for common computations across many areas
 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 "Motifs"
(Use "motif" instead when go from 7 to 13)

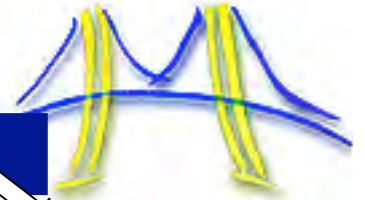
"Motif" Popularity

(Red Hot → Blue Cool)

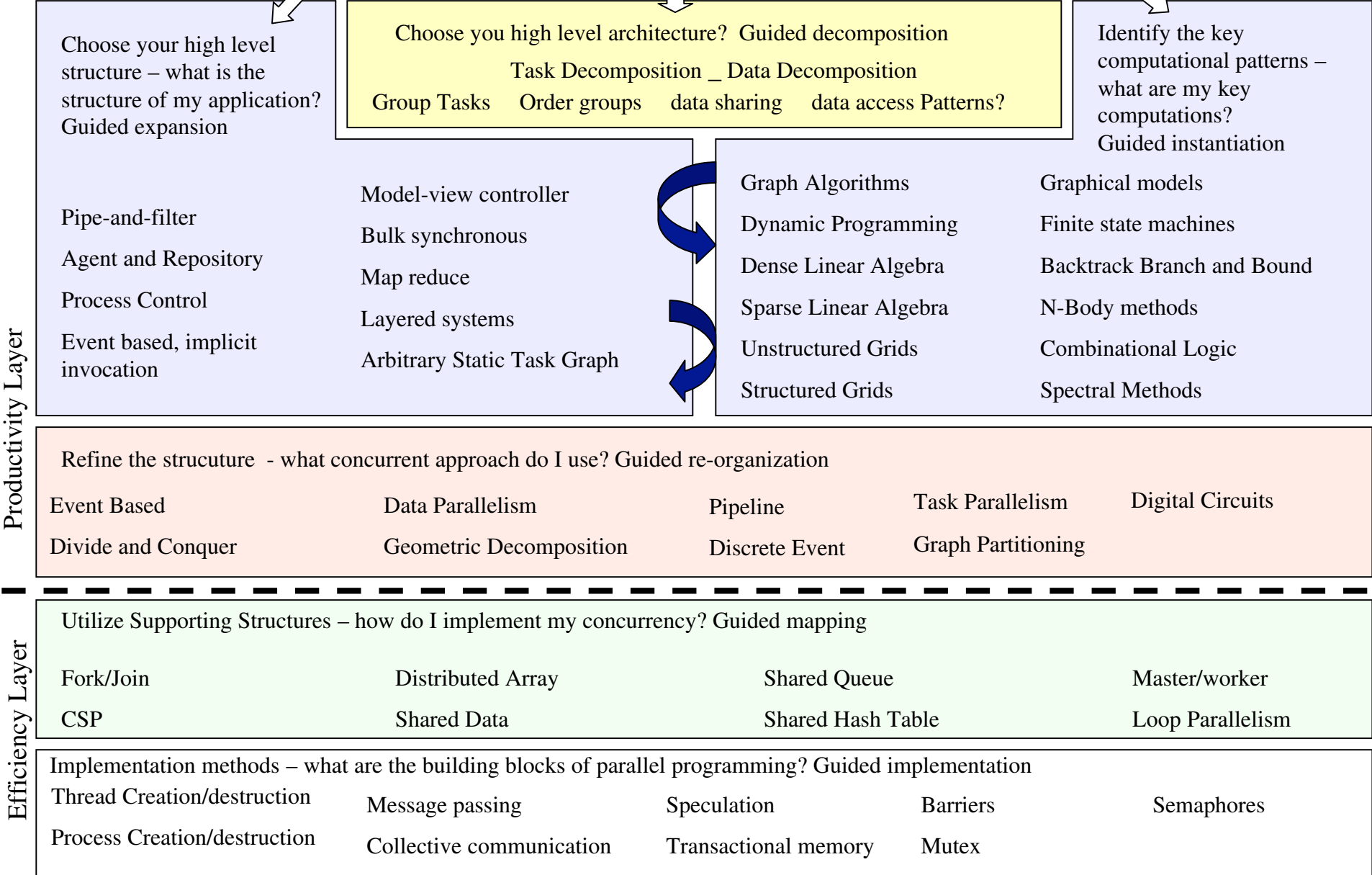


- How do compelling apps relate to 13 motifs?

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



Applications



Productivity Layer

Efficiency Layer

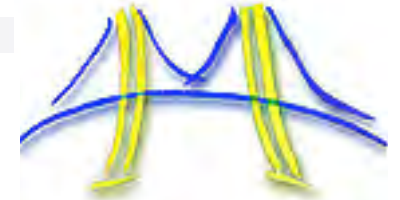


Themes 1 and 2 Summary

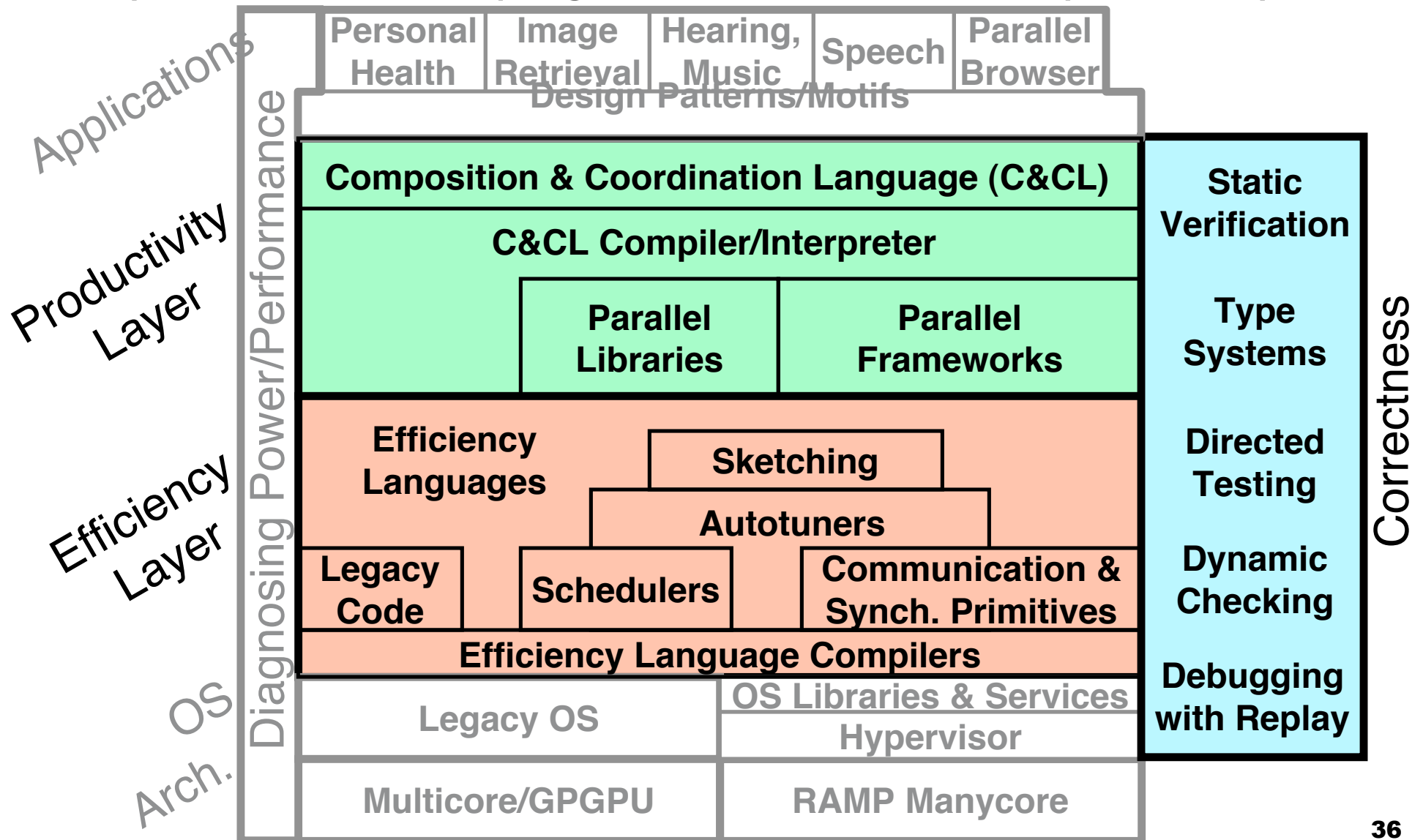


- Application-Driven Research (top down) vs. CS Solution-Driven Research (bottom up)
 - Bet is not that every program speeds up with more cores, but that we can find some compelling ones that do
- Drill down on (initially) 5 app areas to guide research agenda
- Design Patterns + Motifs to guide design of apps through layers

Par Lab Research Overview



Easy to write correct programs that run efficiently on manycore



Theme 3: Developing Parallel SW



- 2 types of programmers \Rightarrow 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 \Rightarrow
Create language for Composition and Coordination (C&C)

C & C Language Requirements (Kathy Yelick)



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

(Koushek Sen)



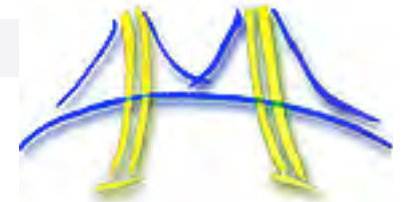
■ Productivity Layer

- 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)

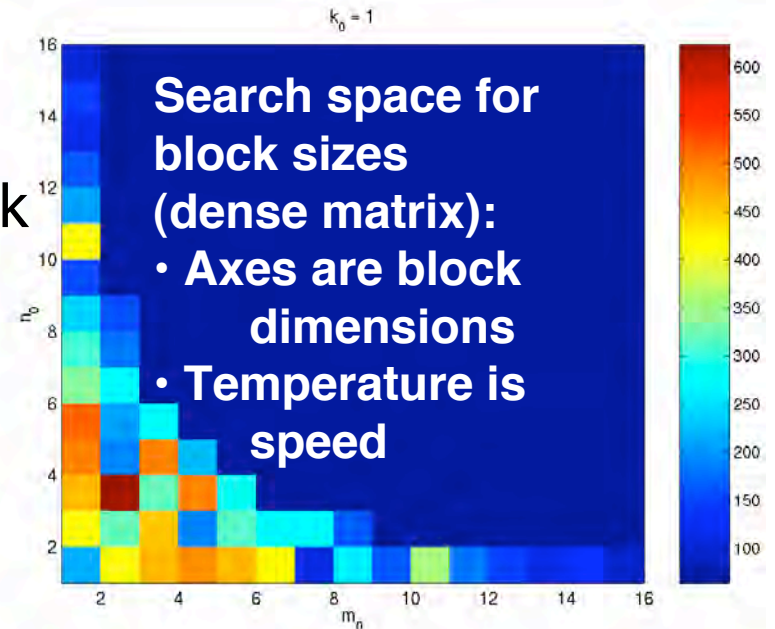
■ Efficiency Layer: Check for subtle concurrency bugs (races, deadlocks, and so on)

- Mixture of verification and automated directed testing
- Error detection on frameworks with sequential code as specification
- Automatic detection of races, deadlocks

21st Century Code Generation (Demmel, Yelick)



- Problem: generating optimal code is like searching for needle in a haystack
- Manycore \Rightarrow 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; Optimizations: BCOO v. BCSR data structures, NUMA, 16b vs. 32b indicies, ...



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, prefetch		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
Base SpMV <small>(median of many matrices)</small>	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

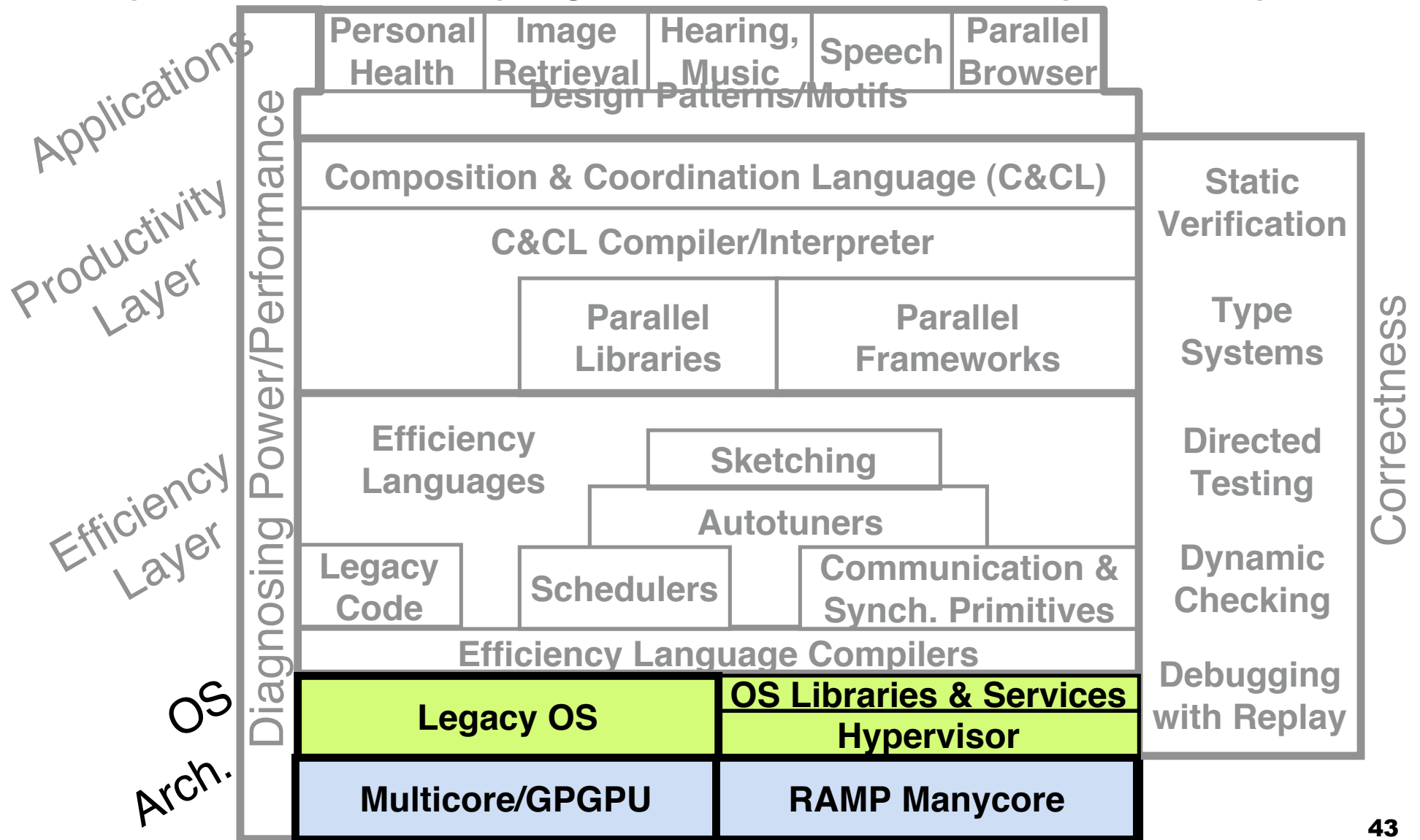


- SpMV: Easier to autotune single local RAM + DMA than multilevel caches + HW and SW prefetching
- Productivity Layer & Efficiency Layer
- C&C Language to compose Libraries/Frameworks
- Libraries and Frameworks to leverage experts

Par Lab Research Overview



Easy to write correct programs that run efficiently on manycore





Theme 4: OS and Architecture

(Krste Asanovic, Eric Brewer, John Kubiawicz)

- 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 OS and architectural support improve productivity, efficiency, and correctness for scalable hardware?
 - Efficiency instead of performance to capture energy as well as performance
- Other HW challenges: power limit, design and verification costs, low yield, higher error rates
- How prototype ideas fast enough to run real SW?



Deconstructing Operating Systems

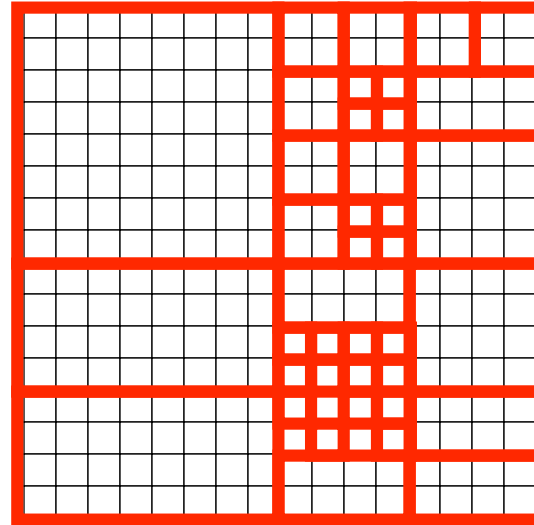


- Resurgence of interest in virtual machines
 - Hypervisor: thin SW layer btw guest OS and HW
- Future OS: libraries where only functions needed are linked into app, on top of thin hypervisor providing protection and sharing of resources
 - Opportunity for OS innovation
- Leverage HW partitioning support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

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 ($\approx 1\text{ns}$)
 - Signals propagate combinatorially
 - Hypervisor sets taps saying where partition sees barrier



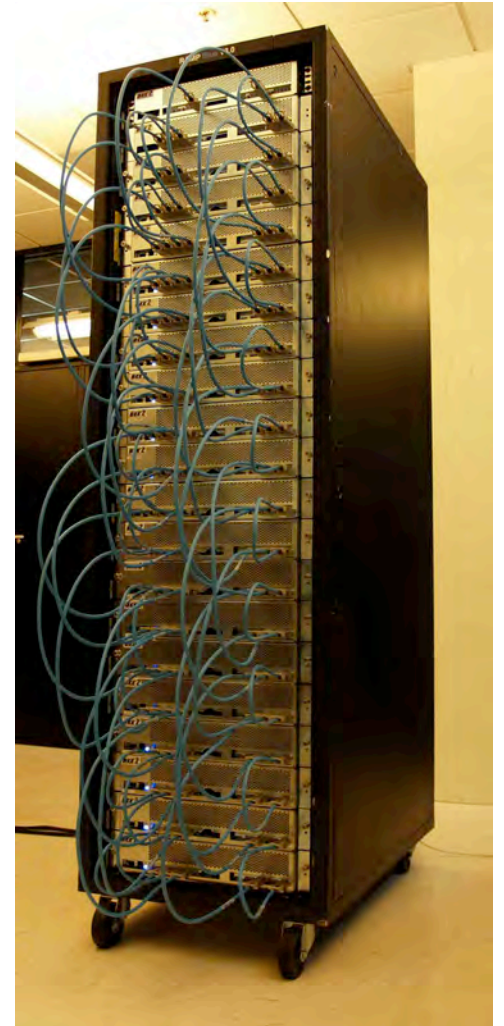
HW Solution: Small is Beautiful

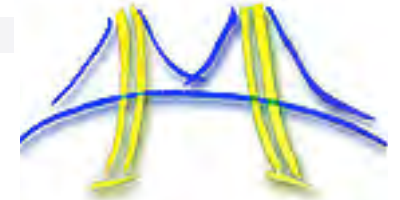
- Want Software Composable Primitives, Not Hardware Packaged Solutions
 - “You’re not going fast if you’re headed in the wrong direction”
 - Transactional Memory is usually a Packaged Solution
- Expect modestly pipelined (5- to 9-stage) CPUs, FPUs, vector, SIMD PEs
 - Small cores not much slower than large cores
- Parallel is energy efficient path to performance: CV^2F
 - Lower threshold and supply voltages lowers energy per op
- 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

1008 Core "RAMP Blue"

(Wawrzynek, Krasnov,... at Berkeley)

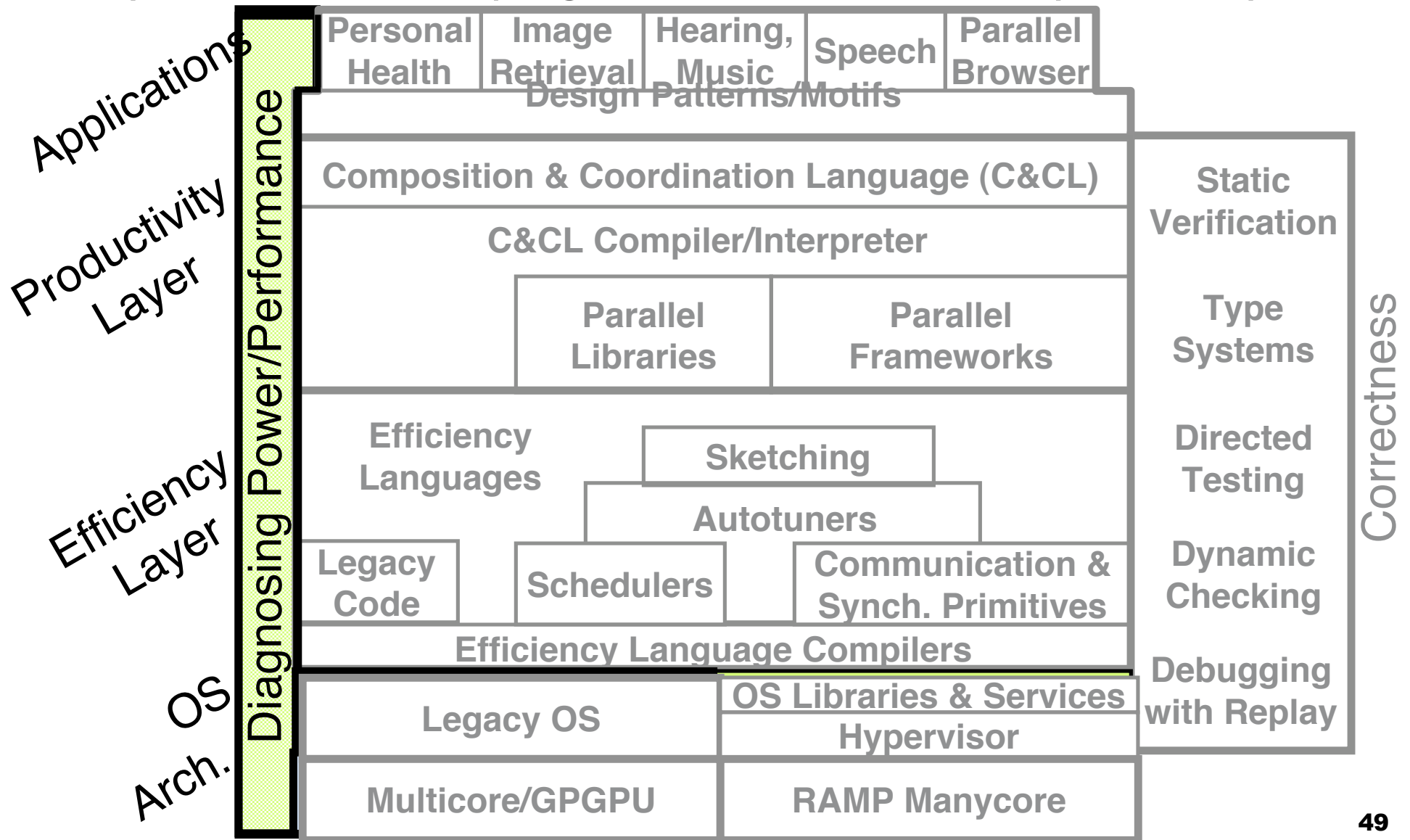
- 1008 = 12 32-bit RISC cores /
FPGA, 4 FGPAs/board, 21 boards
 - Simple MicroBlaze soft cores @ 90 MHz
 - Full star-connection between modules
- NASA Advanced Supercomputing (NAS)
Parallel Benchmarks (all class S)
 - UPC versions (C plus shared-memory abstraction)
CG, EP, IS, MG
- RAMPants creating 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





Par Lab Research Overview

Easy to write correct programs that run efficiently on manycore





Theme 5: Diagnosing Power/ Performance Bottlenecks

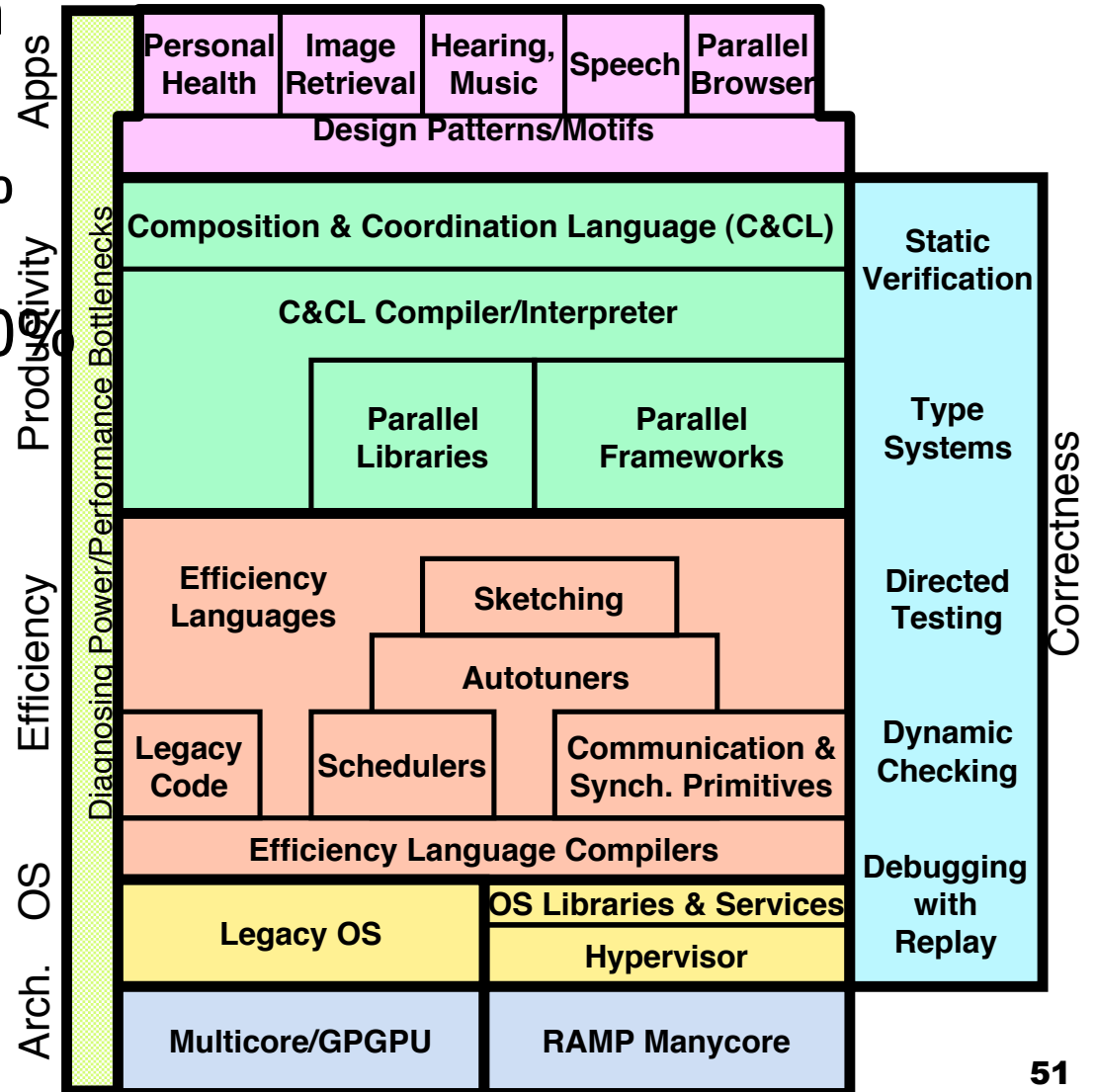
- Collect data on Power/Performance bottlenecks
- Aid autotuner, scheduler, OS in adapting system
- Turn data into useful information that can help efficiency-level programmer improve system?
 - E.g., % peak power, % peak memory BW, % CPU, % network
 - E.g., sample traces of critical paths
- Turn data into useful information that can help productivity-level programmer improve app?
 - Where am I spending my time in my program?
 - If I change it like this, impact on Power/Performance?



Par Lab Summary

- Try Apps-Driven vs. CS Solution-Driven Research
- Design patterns + Motifs
- 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. Compilers
- OS & HW: Primitives vs. Solutions
- Diagnose Power/Perf. bottlenecks

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



Conclusion



- Power wall + Memory Wall = Brick Wall for serial computers
- Industry bet its future on parallel computing, one of the hardest problems in CS
- Once in a career opportunity to reinvent whole hardware/software stack if can make it easy to write correct, efficient, portable, scalable parallel programs
- Failure is not the sin; the sin is not trying.

Acknowledgments





- Intel and Microsoft for being founding sponsors of the Par Lab
- Faculty, Students, and Staff in Par Lab
- **See parlab.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**
- **CACM update (if time permits)**

CACM Rebooted July 2008: to become Best Read Computing Publication?



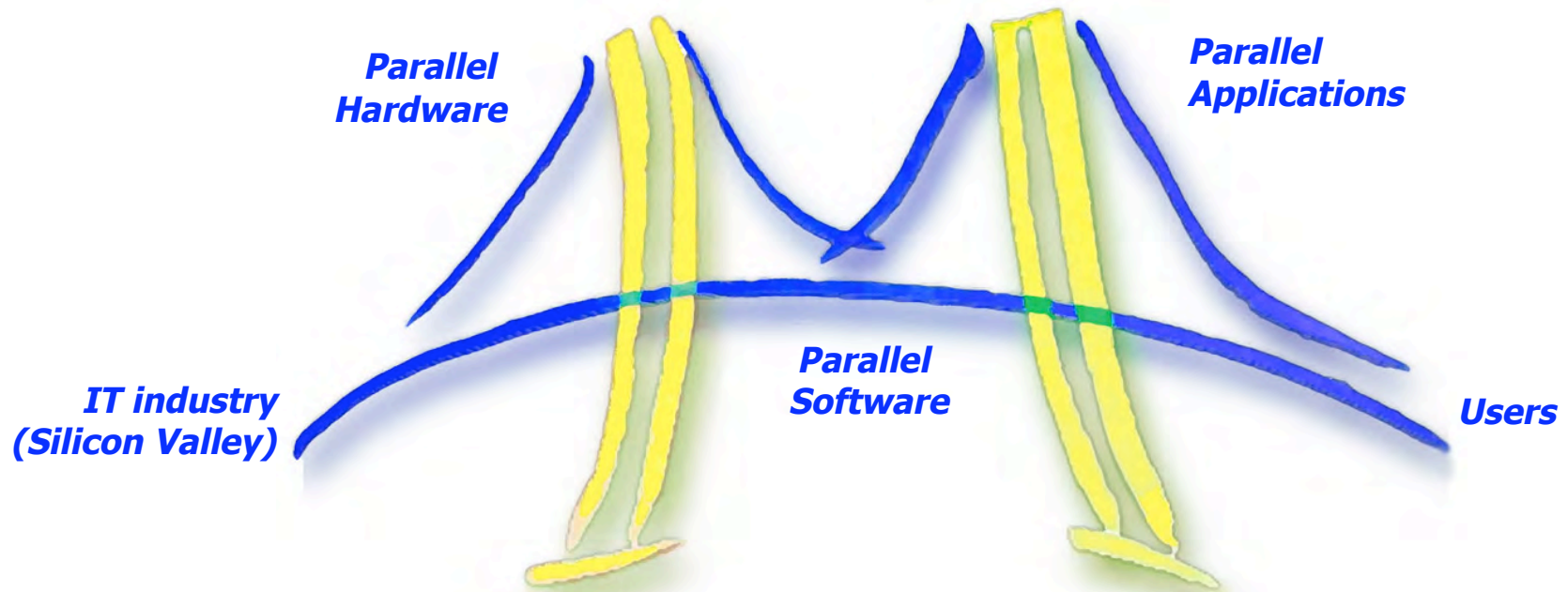
- New direction, editor, editorial board, content
 - Moshe Vardi as EIC + all star editorial board
- 3 News Articles for MS/PhD in CS
 - E.g., "Cloud Computing", "Dependable Design"
- 6 Viewpoints
 - Interview: "The 'Art' of being Don Knuth"
 - "Technology Curriculum for 21st Century": Stephen Andriole (Villanova) vs. Eric Roberts (Stanford)
- 3 Practice articles: Merged *Queue* with *CACM*
 - "Beyond Relational Databases" (Margo Seltzer, Oracle), "Flash Storage" (Adam Leventhal, Sun), "XML Fever"
- 2 Contributed Articles
 - "Web Science" (Hendler, Shadbolt, Hall, Berners-Lee, ...)
 - "Revolution inside the box" (Mark Oskin, Wash.)



(New) CACM is worth reading (again):
Tell your friends!



- 1 Review: invited overview of recent hot topic
 - “Transactional Memory” by J. Larus and C. Kozyrakis
- 2 Research Highlights: Restore field overview?
 - Mine the best of 5000 conferences papers/year: Nominations, then Research Highlight Board votes
 - Emulate *Science* by having 1 page Perspective + 8-page article revised for larger CACM audience
 - “CS takes on Molecular Dynamics” (Bob Colwell) + “Anton, a Special-Purpose Machine for Molecular Dynamics” (Shaw *et al*)
 - “Physical Side of Computing” (Feng Shao) + “The Emergence of a Networking Primitive in Wireless Sensor Networks” (Levis, Brewer, Culler *et al*)



Backup Slides



Utility Computing Arrives?



- Many attempts at utility computing over the years: Sun \$1/CPU hour (2004), HP, IBM, .com collocation SW, ...
- Amazon Elastic Computing Cloud (EC2) / Simple Storage Service (S3) bet is:
 - Low-level platform: Standard VMM/OS, x86 HW
 - Customers store data on S3, install/manage whatever SW they want as VM images on S3
 - NO guarantees of quality, just best effort
 - But very low cost per CPU hour, per GB month, not trivial cost per GB of network traffic

Utility Computing Arrives?



- EC2 CPU: 1 hour of 1000 cores = \$100
 - 1 EC2 Compute Unit
 - ≈ 1.0-1.2 GHz 2007 Opteron or 2007 Xeon core

“Instances”	Platform	Cores	Memory	Disk
Small - \$0.10 / hr	32-bit	1	1.7 GB	160 GB
Large - \$0.40 / hr	64-bit	4	7.5 GB	850 GB – 2 spindles
XLarge - \$0.80 / hr	64-bit	8	15.0 GB	1690 GB – 3 spindles

- Network: Free between S3 and EC2;
External: ≈ \$0.10/GB (in or out)

- Animoto adds application to Facebook

- 25,000 to 250,000 users in 3 days

 - Signing up 20,000 new users per hour at peak

- 50 EC2 instances to 3500 in 2 days

<http://blog.rightscale.com/2008/04/23/animoto-facebook-scale-up/>

<http://www.omnisio.com/v/9ceYTUGdjh9/jeff-bezos-on-animoto>

(Every
16 hours)

