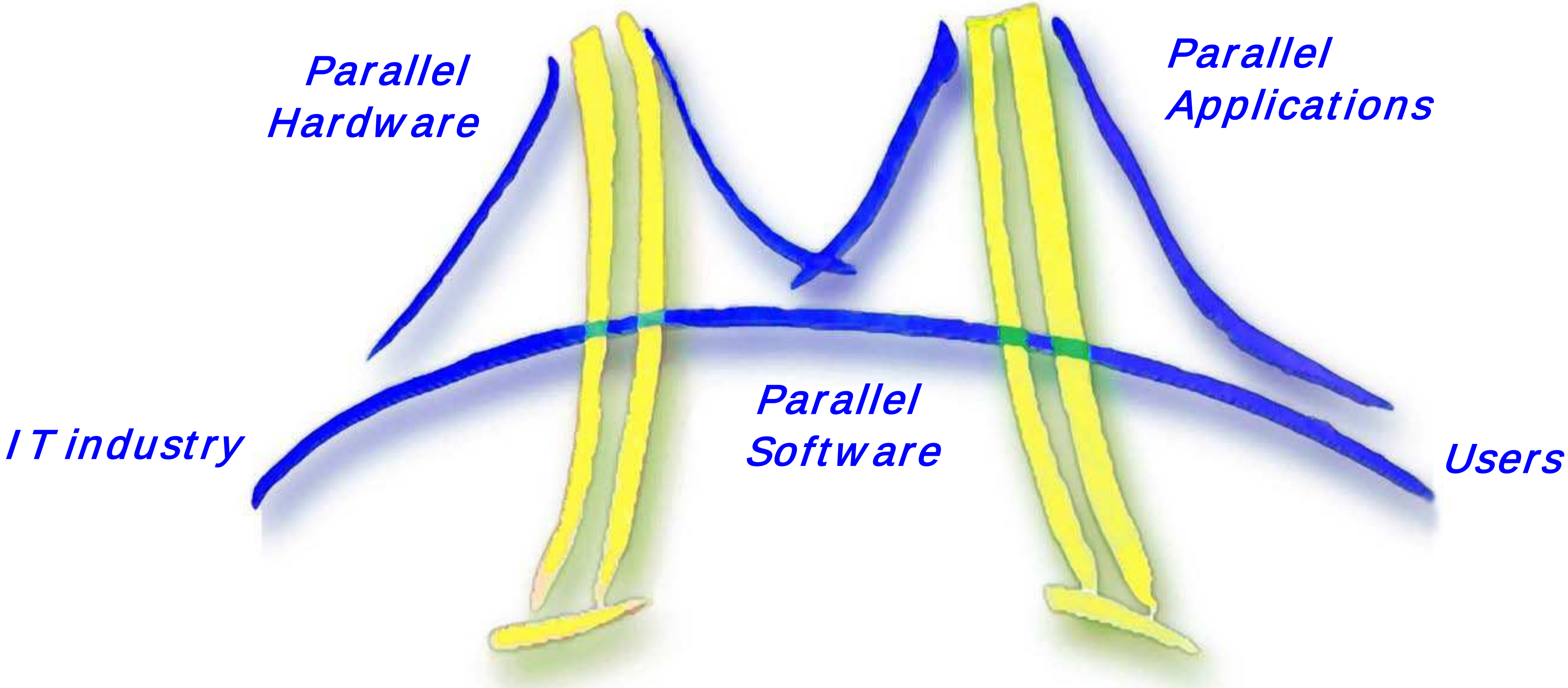


Overview of the UC Berkeley Par Lab

David Patterson
August 2009

A Parallel Revolution, Ready or Not

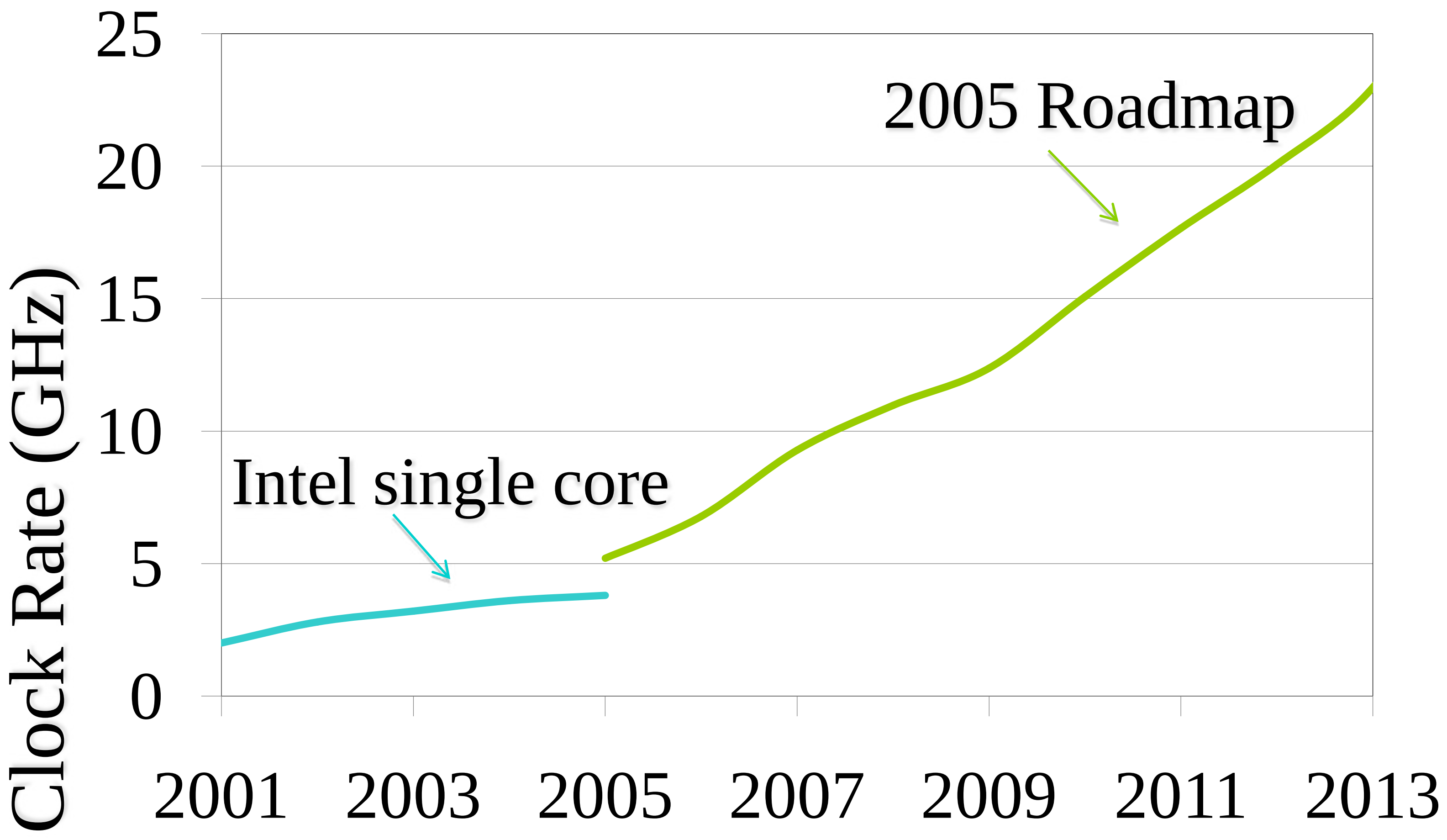
- Power Wall = Brick Wall
  - End of way built microprocessors for last 40 years
  - Intel Pentium 4: most power/transistor inefficient CPU

➔ New "Moore's Law" is 2X processors ("cores") / 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."

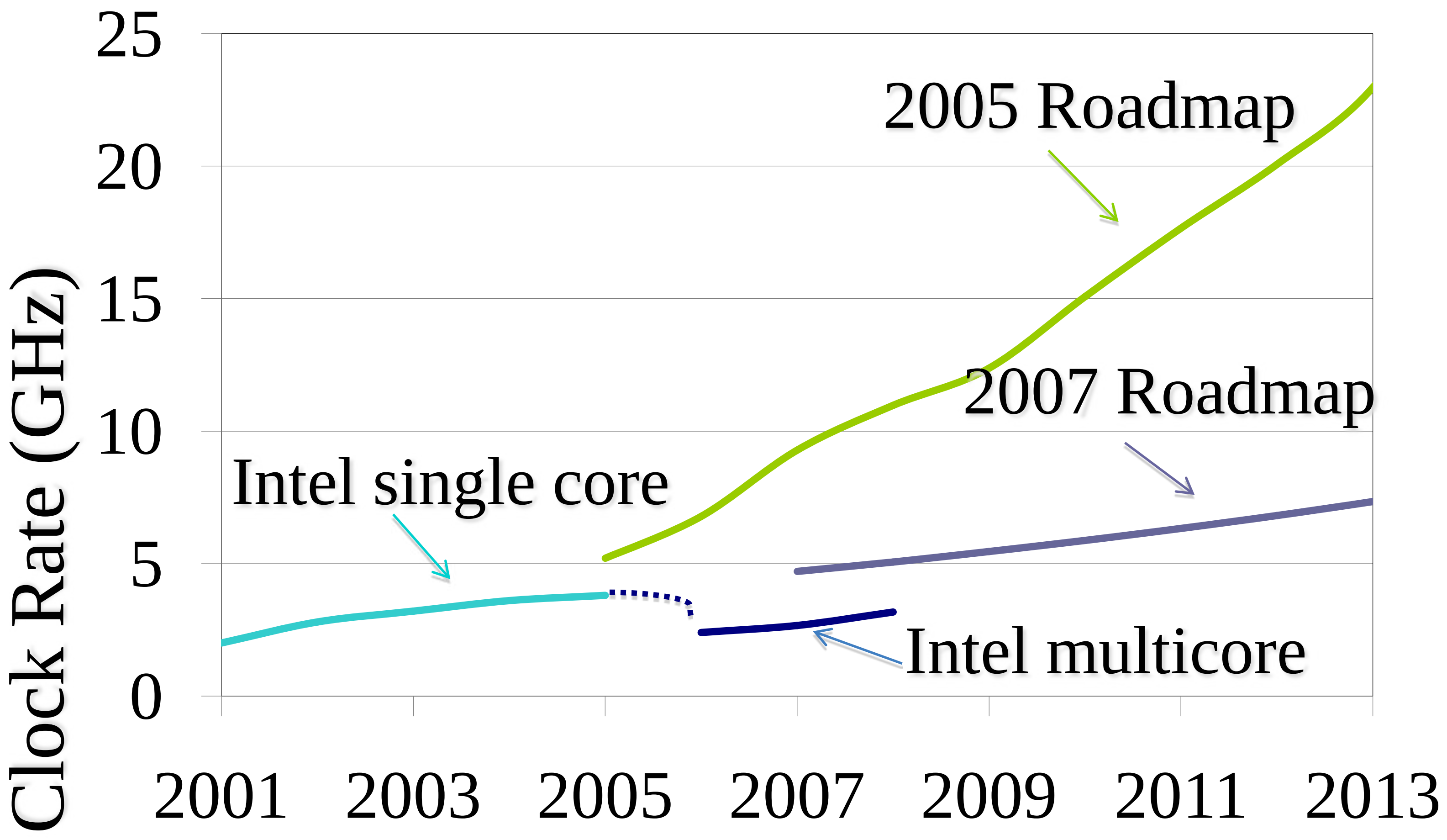
    The Parallel Computing Landscape: A Berkeley View, Dec 2006

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

2005 IT Roadmap Semiconductors



Change in ITS Roadmap in 2 yrs



Why might we succeed this time?

- No Killer Microprocessor to Save Programmers
  - No one is building a faster serial microprocessor
  - For programs to go faster, SW must use parallel HW
- New Metrics for Success vs. Linear Speedup
  - Real Time Latency/Responsiveness and/or MIPS/Joule
  - Just need some new killer parallel apps
    vs. all legacy SW must achieve linear speedup
- Necessity: All the Wood Behind One Arrow
  - Whole industry committed, so more working on it
  - If future growth of IT depends on faster processing at same price (vs. lowering costs like NetBook)

Why might we succeed this time?

- Multicore Synergy with Cloud Computing
  - Cloud Computing apps parallel even if client not parallel
- Vitality of Open Source Software
  - OSS community more quickly embraces advances?
- Single-Chip Multiprocessors Enable Innovation
  - Enables inventions that were impractical or uneconomical when multiprocessors were 100s chips
- FPGA prototypes shorten HW/SW cycle
  - Fast enough to run whole SW stack, can change every day vs. every 4 to 5 years when do chips

Need a Fresh Approach to Parallelism

- Past parallel projects often dominated by hardware/architecture
  - This is the one true way to build computers: software must adapt to this breakthrough
  - ILLIAC IV, Thinking Machines CM-2, Transputer, Kendall Square KSR-1, Silicon Graphics Origin 2000 …
- Or sometimes by programming language
  - This is the one true way to write programs: hardware must adapt to this breakthrough
  - ID, Backus Functional Language FP, Occam, Linda, High Performance Fortran, Chapel, X10, Fortress …
- Apps usually an afterthought

Need a Fresh Approach to Parallelism

- Berkeley researchers from many backgrounds started meeting in Feb. 2005 to discuss parallelism
  - Krste Asanovic, Ras Bodik, Jim Demmel, Kurt Keutzer, John Kubiatowicz, 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 core increase every 2 years (!)

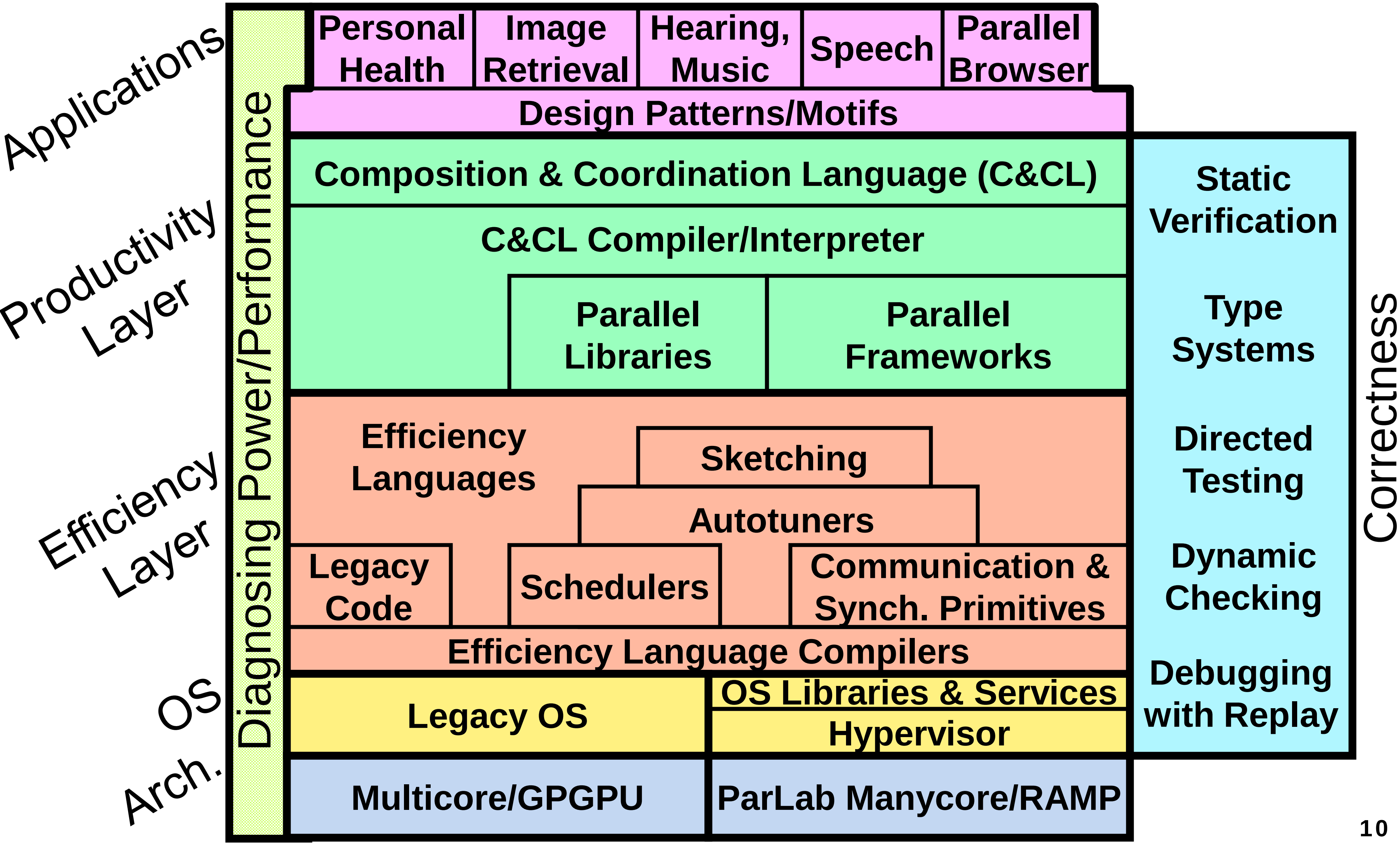
Par Lab's original research "bets"

- Let compelling applications drive research agenda
- Software platform: data center + mobile client
- Identify common programming patterns
- Productivity versus efficiency programmers
- Autotuning and software synthesis
- Build correctness + power/performance diagnostics into stack
- OS/Architecture support applications, provide primitives not pre-packaged solutions
- FPGA simulation of new parallel architectures: RAMP

Above all, no preconceived big idea – see what works driven by application needs

Par Lab Research Overview

Easy to write correct programs that run efficiently on manycore



Dominant Application Platforms

- Data Center or Cloud (“Server”)
- Laptop/Handheld (“Mobile Client”)
- Both together (“Server+Client”)
  - New ParLab-RADLab collaborations
- Par Lab focuses on mobile clients
  - But many technologies apply to data center

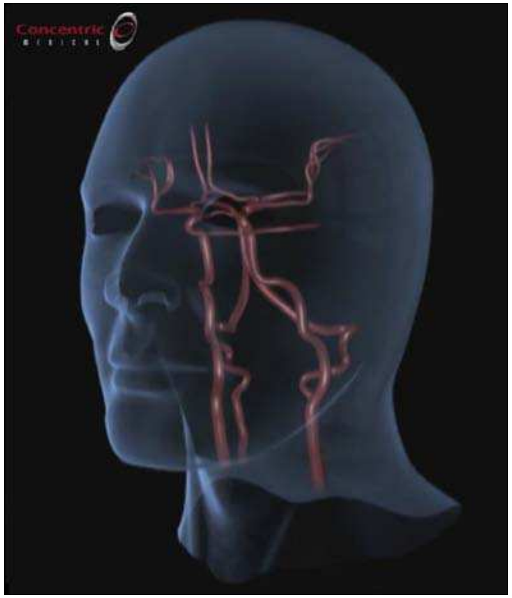
Music and Hearing Application
(David Wessel)

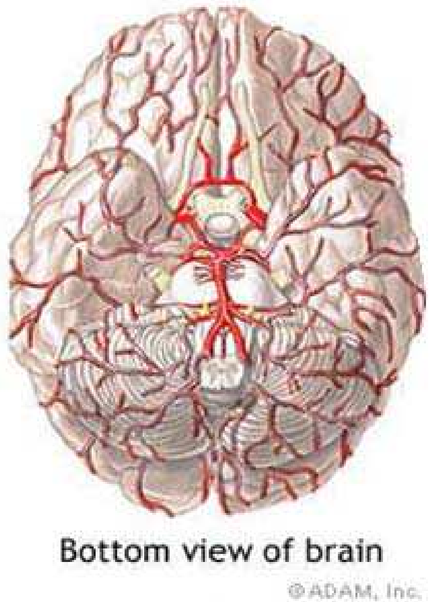
- Musicians have an insatiable appetite for computation + real-time demands
  - More channels, instruments, more processing, more interaction!
  - Latency must be low (5 ms)
  - Must be reliable (No clicks!)

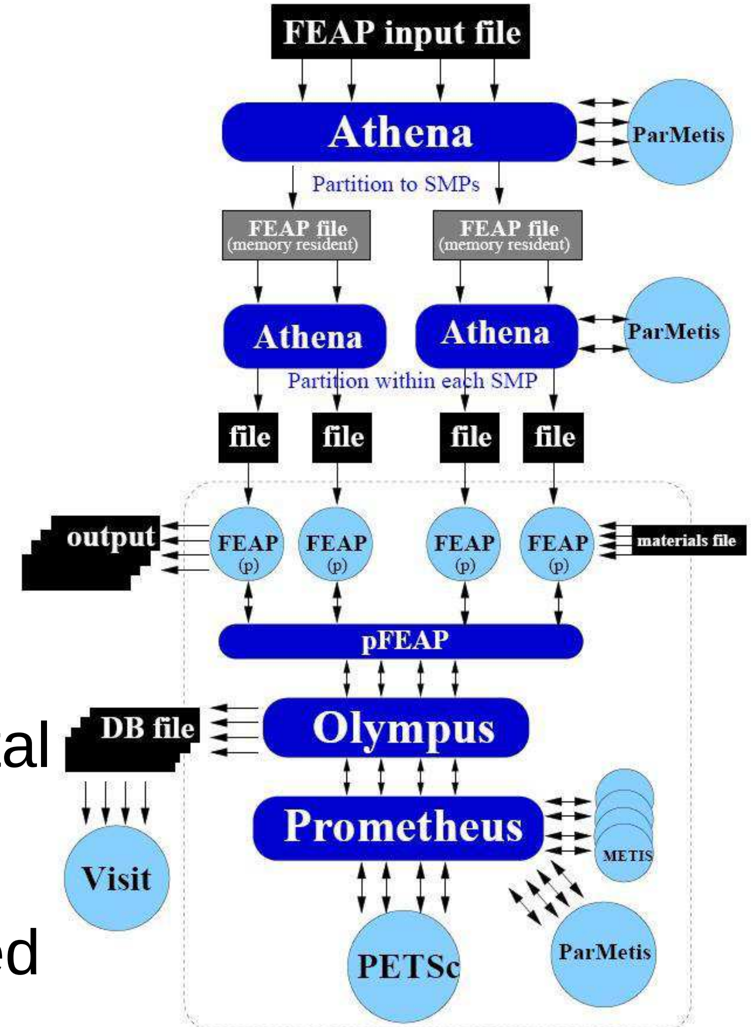
1. Music Enhancer
   - Enhanced sound delivery systems for home sound systems using large microphone and speaker arrays
   - Laptop/Handheld recreate 3D sound over ear buds
2. Hearing Augmenter
   - Handheld as accelerator for hearing aid
3. Novel Instrument User Interface
   - New composition and performance 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.

Health Application: Stroke Treatment (Tony Keaveny)



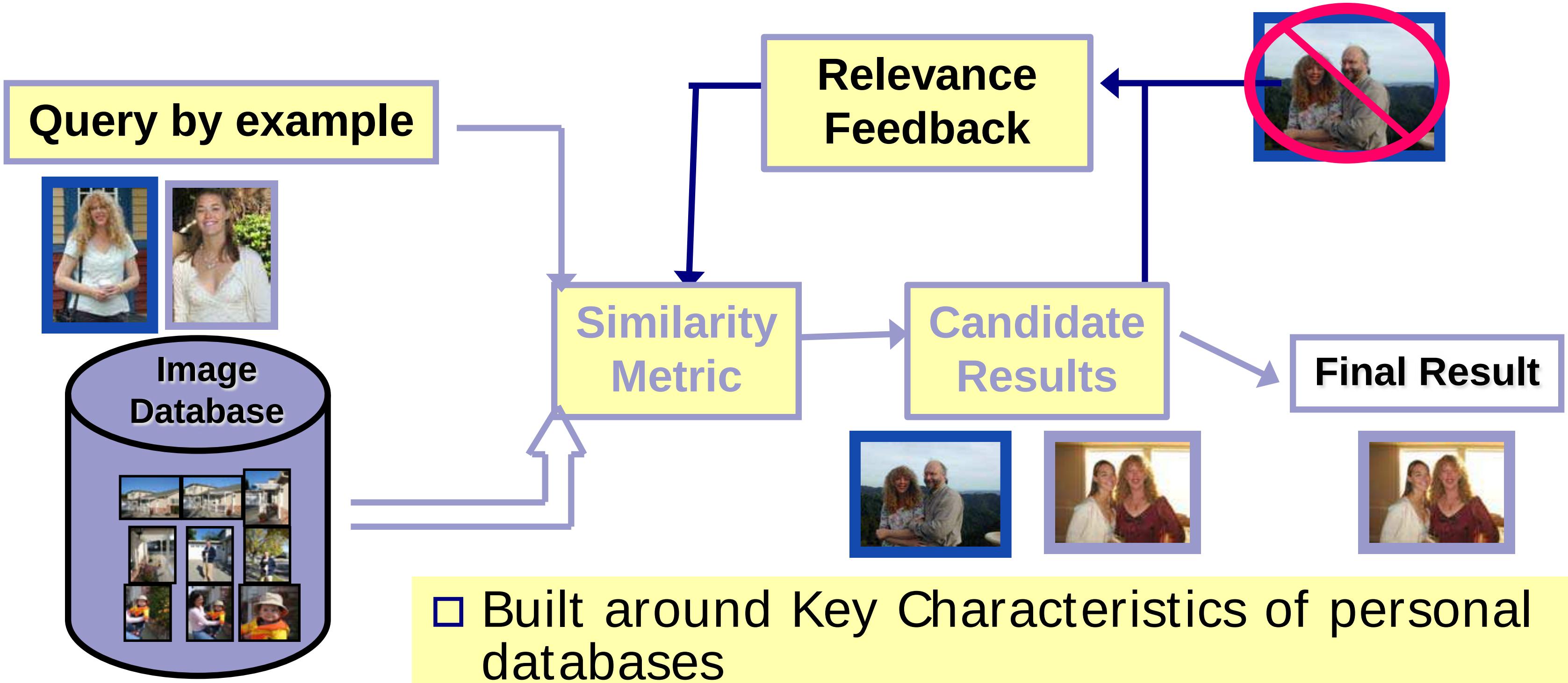




- Stroke treatment time-critical, need supercomputer performance in hospital
- Goal: First true 3D Fluid-Solid Interaction analysis of Circle of Willis
- Based on existing codes for distributed clusters

Content-Based Image Retrieval

(Kurt Keutzer)



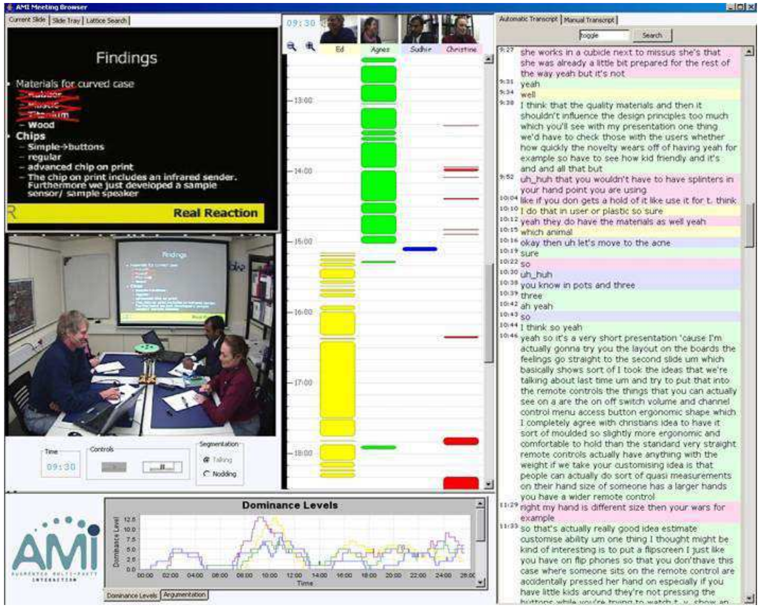
1000's of images

- 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

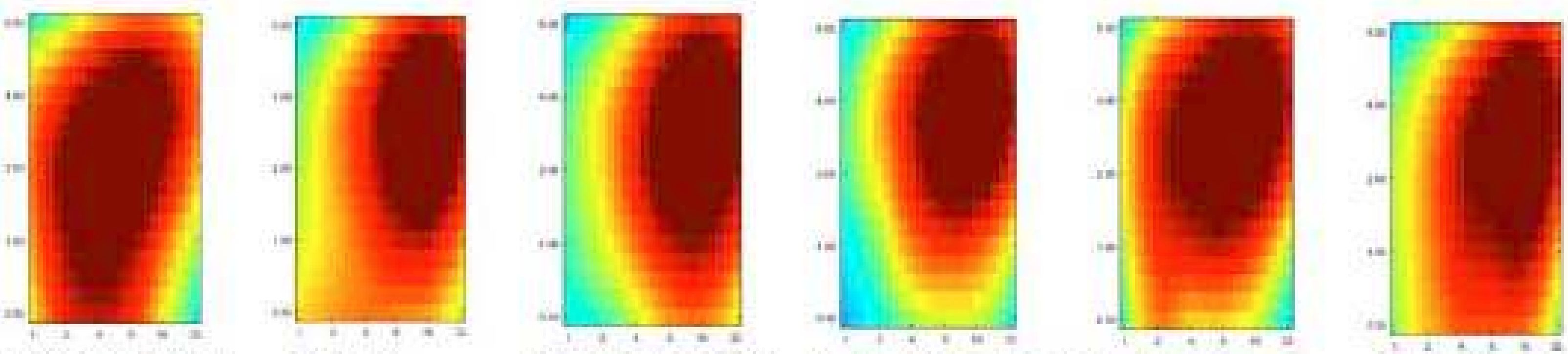
Robust Speech Recognition

(Nelson Morgan)

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



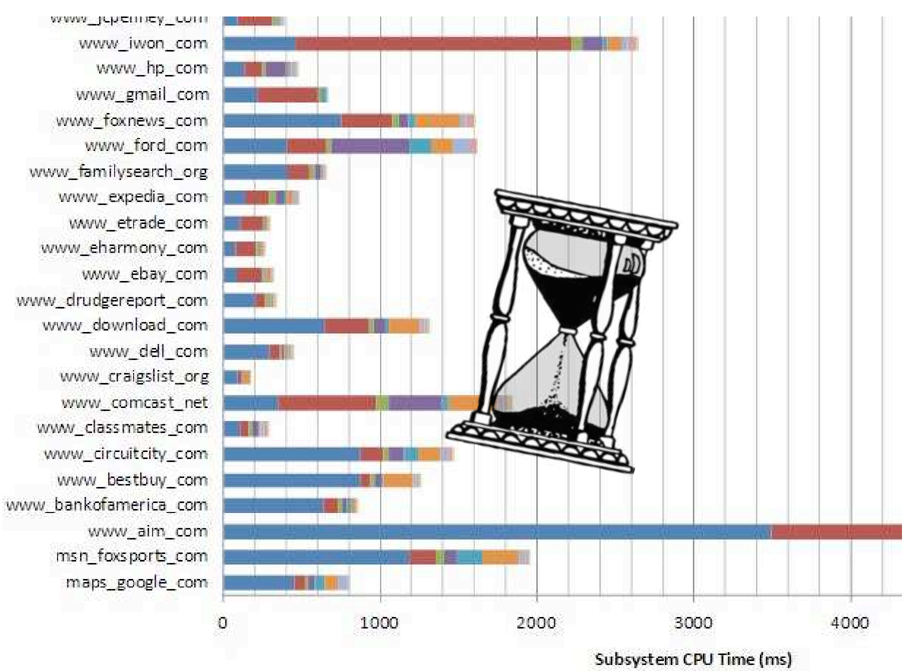
- Use cortically-inspired manystream spatio-temporal features to tolerate noise

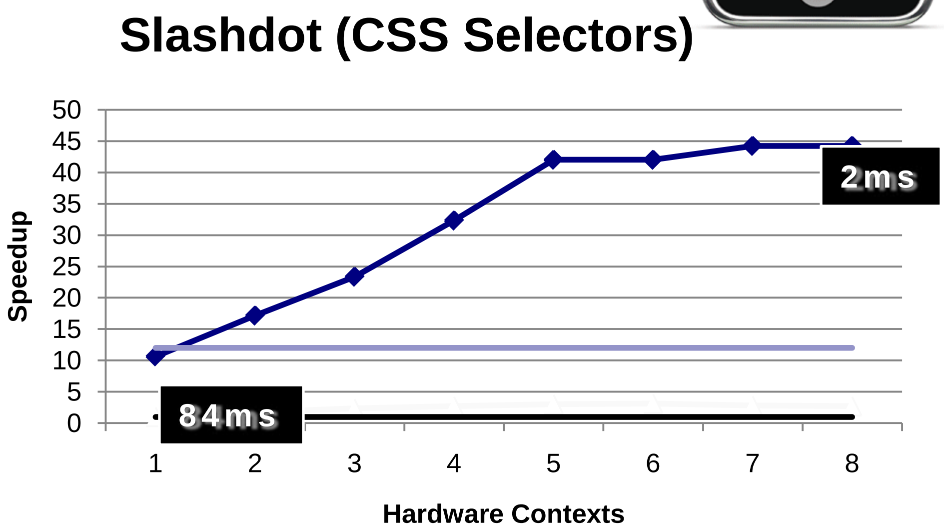


Speech babble | Factory noise | F16 Interior noise | Car Interior noise | Tank interior noise | Battleship noise

Parallel Browser
(Ras Bodik)

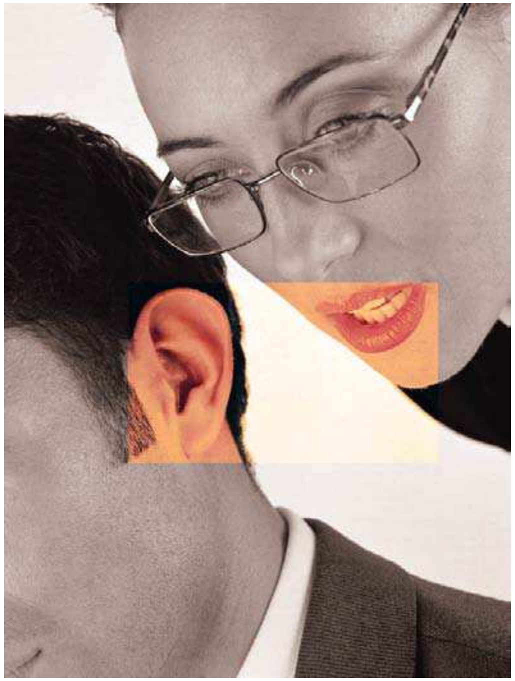
- Goal: Desktop quality browsing on handhelds
  - Enabled by 4G networks, better output devices
- Bottlenecks to parallelize
  - Parsing, Rendering, Scripting





Compelling Apps in a Few Years

- Name Whisperer
  - Built from Content Based Image Retrieval
  - Like Presidential Aid
- Handheld scans face of approaching person
- Matches image database
- Whispers name in ear, along with how you know him



Architecting Parallel Software with Patterns (Kurt Keutzer/Tim Mattson)

Our initial survey of many applications brought out common recurring patterns:

"Dwarfs" -> Motifs

- Computational patterns
- Structural patterns

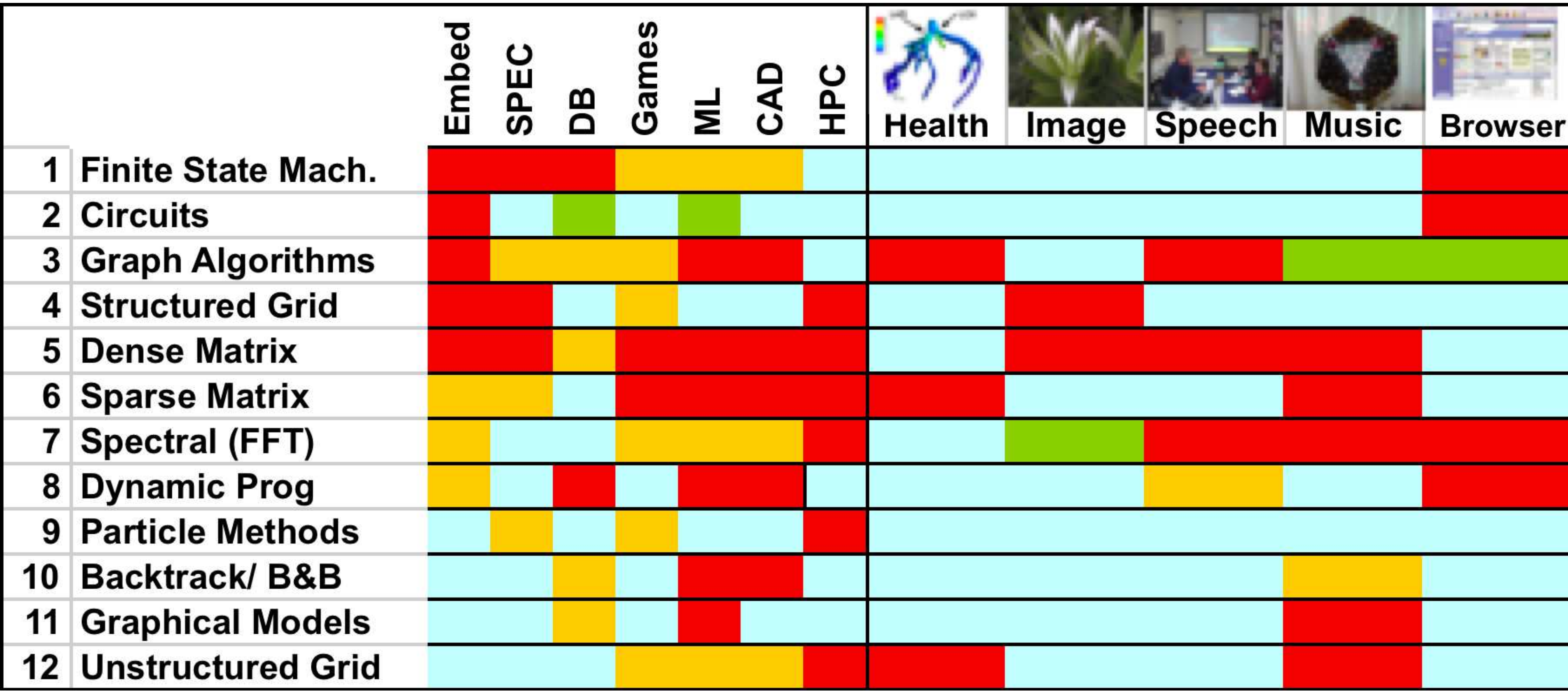
Insight: Successful codes have a comprehensible software architecture:

- Patterns give human language in which to describe architecture

Motif (nee "Dwarf") Popularity

(Red Hot ⟨ Blue Cool)

- How do compelling apps relate to 12 motifs?



Architecting Parallel Software

Decompose Tasks/Data

Order tasks **Identify Data Sharing and Access**

Identify the Software Structure

- •Pipe-and-Filter
- •Agent-and-Repository
- •Event-based
- •Bulk Synchronous
- •MapReduce
- •Layered Systems
- •Arbitrary Task Graphs

Identify the Key Computations

- Graph Algorithms
- Dynamic programming
- Dense/Spare Linear Algebra
- (Un)Structured Grids
- Graphical Models
- Finite State Machines
- Backtrack Branch-and-Bound
- N-Body Methods
- Circuits
- Spectral Methods

Productivity/Efficiency and Patterns

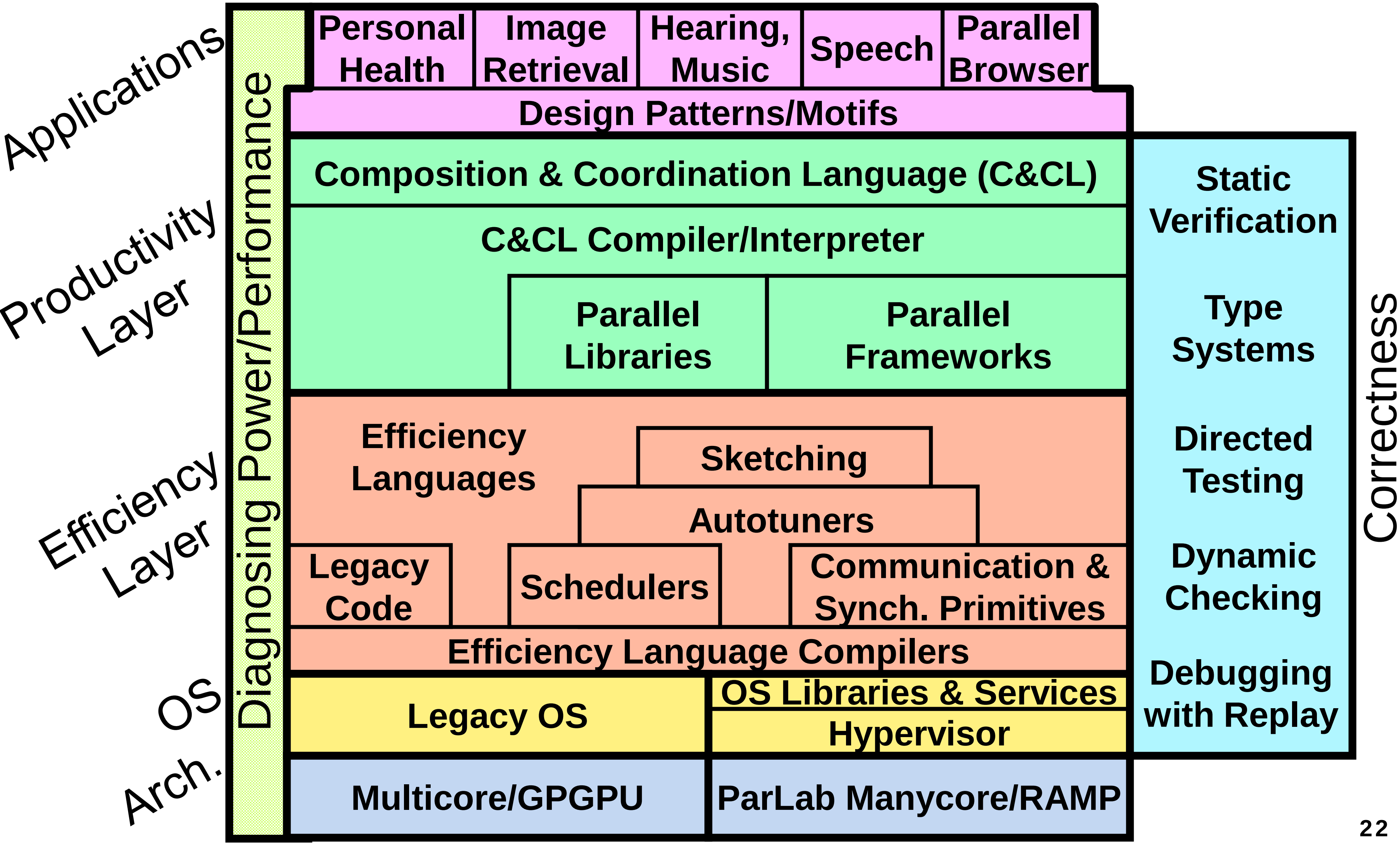
1. Domain Experts + Application patterns & frameworks → End-user, application programs
2. Domain-literate programming gurus (1% of the population). + Parallel patterns & programming frameworks → Application frameworks
3. Parallel programming gurus (1-10% of programmers) → Parallel programming frameworks

The hope is for Domain Experts to create parallel code with little or no understanding of parallel programming.

Leave hardcore "bare metal" efficiency-layer programming to the parallel programming experts

Par Lab Research Overview

Easy to write correct programs that run efficiently on manycore



Par Lab is Multi-Lingual

- Applications require ability to compose parallel code written in many languages and several different parallel programming models
  - Let application writer choose language/model best suited to task
  - High-level productivity code and low-level efficiency code
  - Old legacy code plus shiny new code
- Correctness through all means possible
  - Static verification, annotations, directed testing, dynamic checking
  - Framework-specific constraints on non-determinism
  - Programmer-specified semantic determinism
  - Require common spec between languages for static checker
- Common linking format at low level (Lithe) not intermediate compiler form
  - Support hand-tuned code and future languages & parallel models

Selective Embedded Just-In-Time Specialization (SEJITS) for Productivity

- Modern scripting languages (e.g., Python and Ruby) have powerful language features and are easy to use
- Idea: Dynamically generate source code in C within the context of a Python or Ruby interpreter, allowing app to be written using Python or Ruby abstractions but automatically generating, compiling C at runtime
- Like a JIT but
  - **Selective**: Targets a particular method and a particular language/platform (C+OpenMP on multicore or CUDA on GPU)
  - **Embedded**: Make specialization machinery productive by implementing in Python or Ruby itself by exploiting key features: introspection, runtime dynamic linking, and foreign function interfaces with language-neutral data representation

Selective Embedded Just-In-Time Specialization for Productivity

- Case Study: Stencil Kernels on AMD Barcelona, 8 threads
- Hand-coded in C+OpenMP: 2-4 days
- SEJITS in Ruby: 1-2 hours
- Time to run 3 stencil codes:

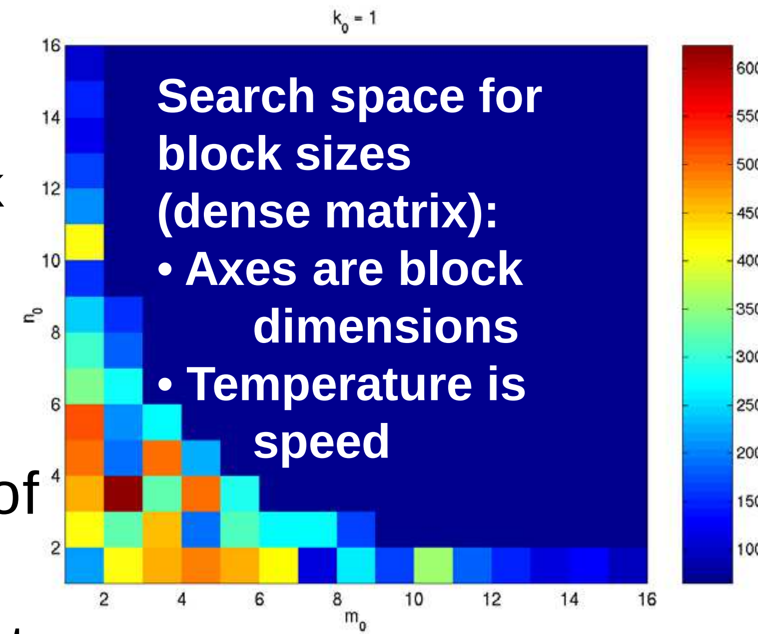
Hand-coded (seconds)	SEJITS from cache (seconds)	Extra JIT-time 1st time executed (seconds)
0.74	0.74	0.25
0.72	0.70	0.27
1.26	1.26	0.27

Recent Results: Active Testing

- Pallavi Joshi, Chang-Seo Park,
  - Advisor Koushik Sen
- Problem: Concurrency Bugs
- Actively control the scheduler to force potentially buggy schedules: Data races, Atomicity Violations, Deadlocks
- Found parallel bugs in real OSS code: Apache Commons Collections, Java Collections Framework, Java Swing GUI framework, and Java Database Connectivity (JDBC)

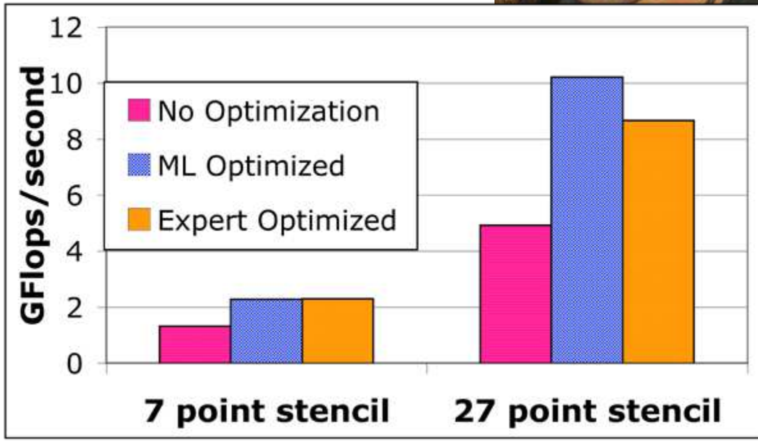
Autotuning for Code Generation (Demmel, Yelick)

- Problem: generating optimal code like searching for needle in 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)

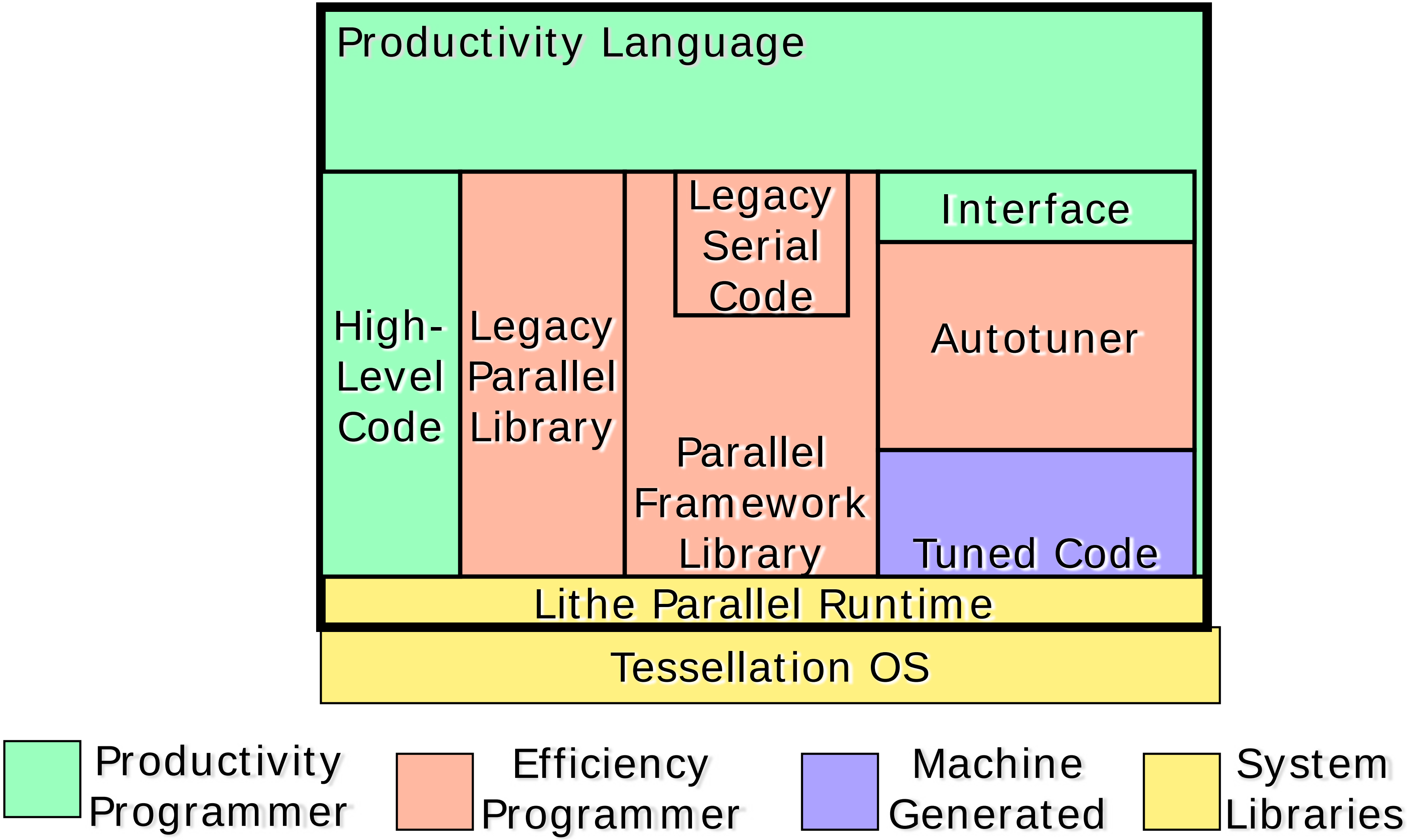


Results: Making Autotuning "Auto"

- Archana Ganapathi & Kaushik Datta
  - Advisors Jim Demmel and David Patterson
- Problem: need expert in architecture *and* algorithm for search heuristics
- Instead, Machine Learning to Correlate Optimizations and Performance
- Evaluate in 2 hours vs. 6 months
- Match or Beat Expert for Stencil Dwarfs



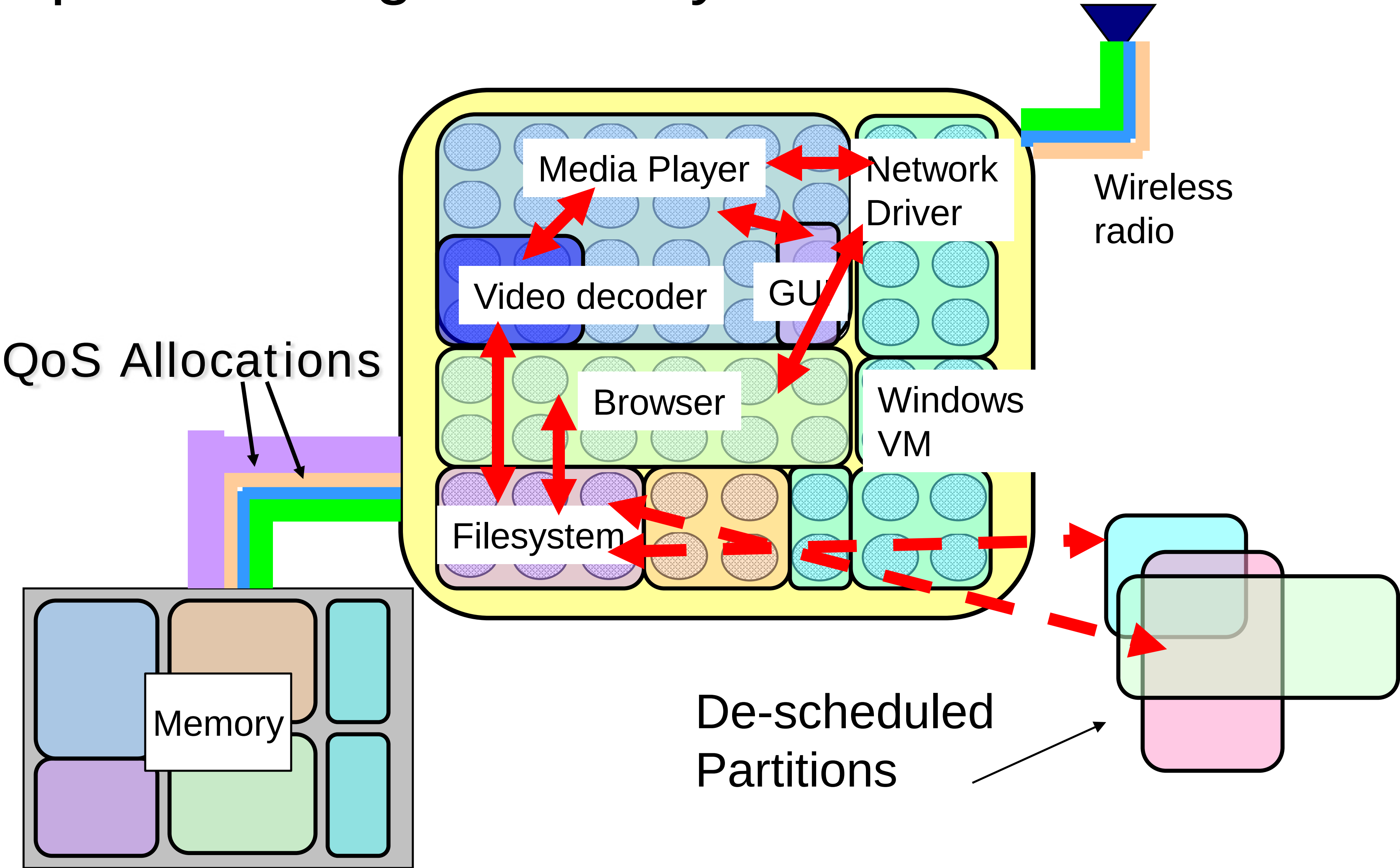
Anatomy of a Par Lab Application



From OS to User-Level Scheduling

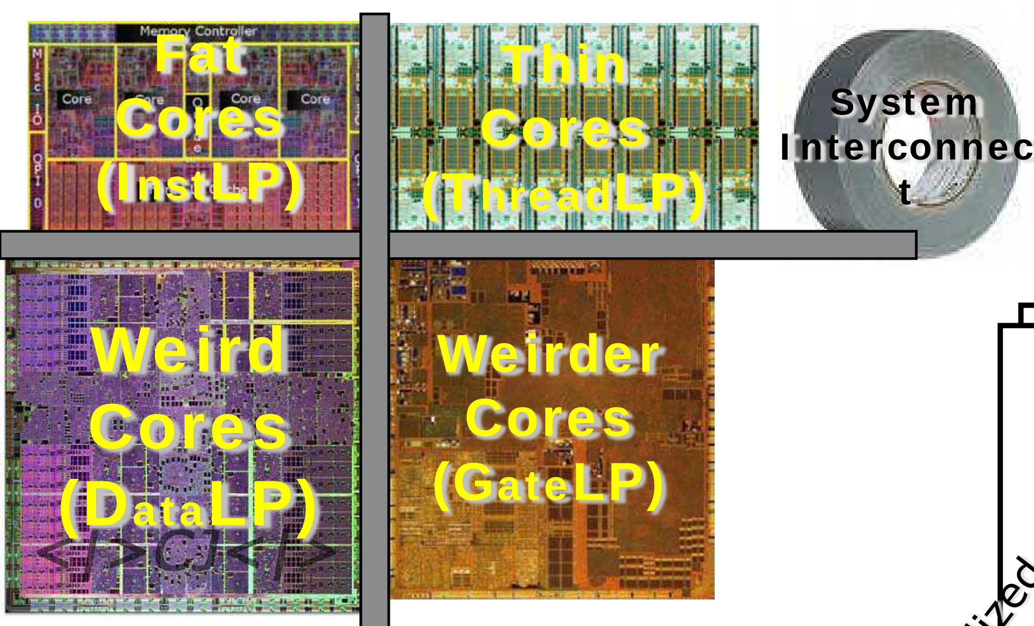
- Tessellation OS allocates hardware resources (e.g., cores) at coarse-grain, and user software shares hardware threads co-operatively using Lithe ABI
- Lithe provides performance composability for multiple concurrent and nested parallel libraries
  - Already supports linking of parallel OpenMP code with parallel TBB code, without changing legacy OpenMP/TBB code and without measurable overhead

Tessellation: Space-Time partitioning for manycore client OS



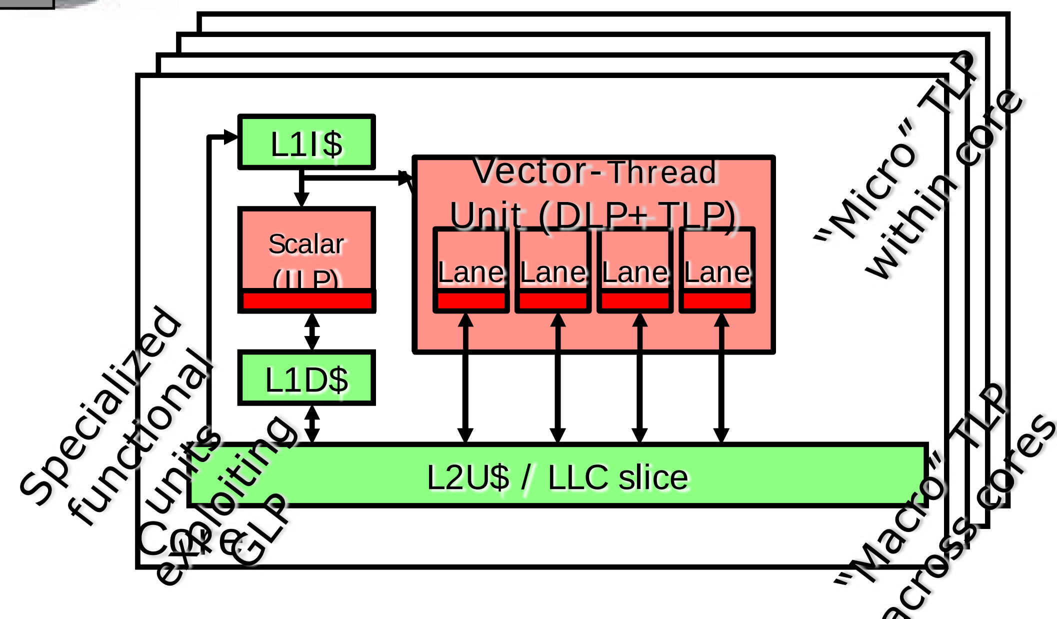
Par Lab Architecture

- Create a long-lived horizontal software platform for independent software vendors (ISVs)
  - ISVs won't rewrite code for each chip or system
  - Customer buys application from ISV 8 years from now, wants to run on machine bought 13 years from now (and see improvements)



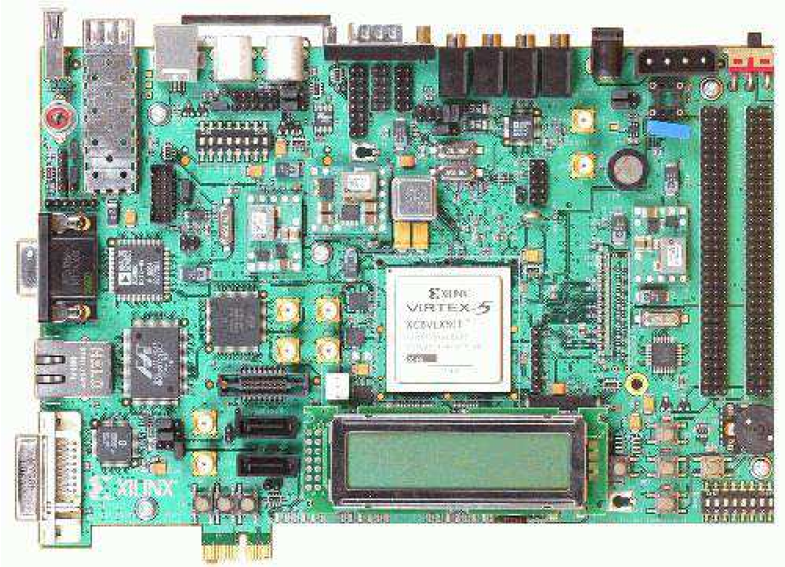
Not multiple paradigms of core

…instead, one type of multi-paradigm core



Recent Results: RAMP Gold

- Rapid accurate simulation of manycore architectural ideas using FPGAs
- Initial version models 64 cores of SPARC v8 with shared memory system on $750 board



	Cost	Performance (MIPS)	Simulations per day
Software Simulator	$2,000	0.1 - 1	1
RAMP Gold	$2,000 + $750	50 - 100	100

New Par Lab: Opened Dec 1, 2008

- 5th Floor South Soda Hall
- Founding Partners: Intel and Microsoft
  - Affiliates: National Instr., NEC, Nokia, Nvidia, Samsung

Recent Results: App Acceleration

- Bryan Catanzaro: Parallelizing Computer Vision (image segmentation) using GPU
- Problem: Malik's highest quality algorithm is 7.8 minutes / image on a PC
- Invention + talk within Par Lab on parallelizing phases using new algorithms, data structures
  - Bor-Yiing Su, Yunsup Lee, Narayanan Sundaram, Mark Murphy, Kurt Keutzer, Jim Demmel, and Sam Williams
- Current GPU result: 2.1 seconds / image
- > 200X speedup
  - Factor of 10 quantitative change is a qualitative change
- Malik: "This will revolutionize computer vision."

Par Lab's original research "bets"

- Let compelling applications drive research agenda
- Software platform: data center + mobile client
- Identify common programming patterns
- Productivity versus efficiency programmers
- Autotuning and software synthesis
- Build correctness + power/perf. diagnostics into stack
- OS/Architecture support applications, provide primitives not pre-packaged solutions
- FPGA simulation of new parallel architectures: RAMP

Above all, no preconceived big idea –
see what works driven by application needs

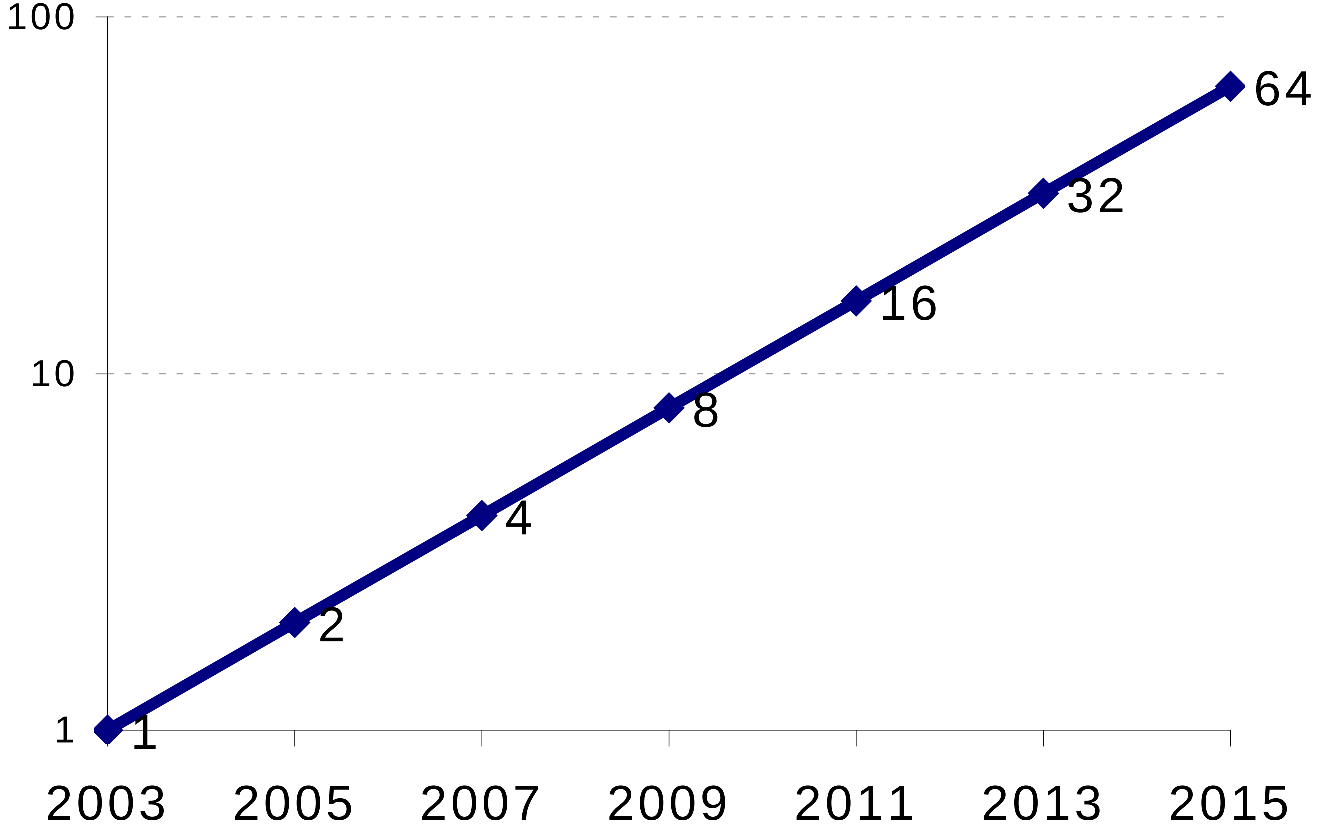
- To learn more: http://parlab.eecs.berekeley.edu

Acknowledgments

- Faculty, Students, and Staff in Par Lab
- Intel, Microsoft Par Lab founding sponsors.
  National Instr., NEC, Nokia, Nvidia Samsung affiliates
  - Contact me if interested in becoming Par Lab Affiliate (pattrsn@cs.berkeley.edu)
- **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**

University Target 8 cores or 100s?

- 5-year research project aimed +8 year technology?
- 2X cores per technology generation



- 64 cores/chip in 2015 is conventional wisdom