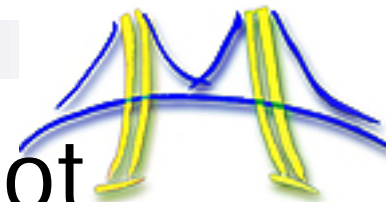


## Par Lab Overview

Dave Patterson  
Parallel Computing Laboratory (Par Lab)  
U.C. Berkeley  
February 2009

# A Parallel Revolution, Ready or Not



- Power Wall = **Brick Wall**

- ⇒ End of way built microprocessors for last 40 years

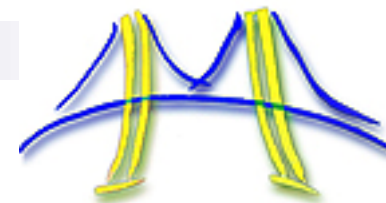
- ➔ New Moore's Law is 2X processors ("cores") per chip every technology generation, but  $\approx$  same clock rate

- "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

# 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 Neda, 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 (!)

# Context: Re-inventing Client/Server

- “The Datacenter is the Computer”

- Building sized computers: AWS, Google, MS, ...
- Private and Public



- “The Laptop/Handheld is the Computer”

- '07: Number HP laptops > desktops
- 1B+ Cell phones/yr, increasing in function



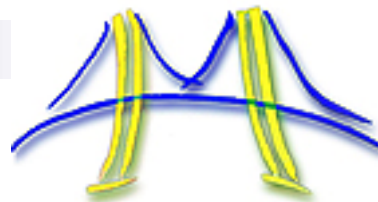
- Otellini demoed “Universal Communicator”

- Combination cell phone, PC and video device

- Apple iPhone, Android, Windows Mobile

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





## 5 Themes of Par Lab

### 1. Applications

- Compelling apps drive top-down research agenda

### 2. Identify Common Design Patterns and “Bricks”

Breaking through disciplinary boundaries

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

2 Layers + Coordination & Composition Language  
+ Autotuning

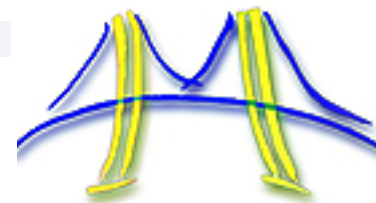
### 4. OS and Architecture

Composable primitives, not packaged solutions

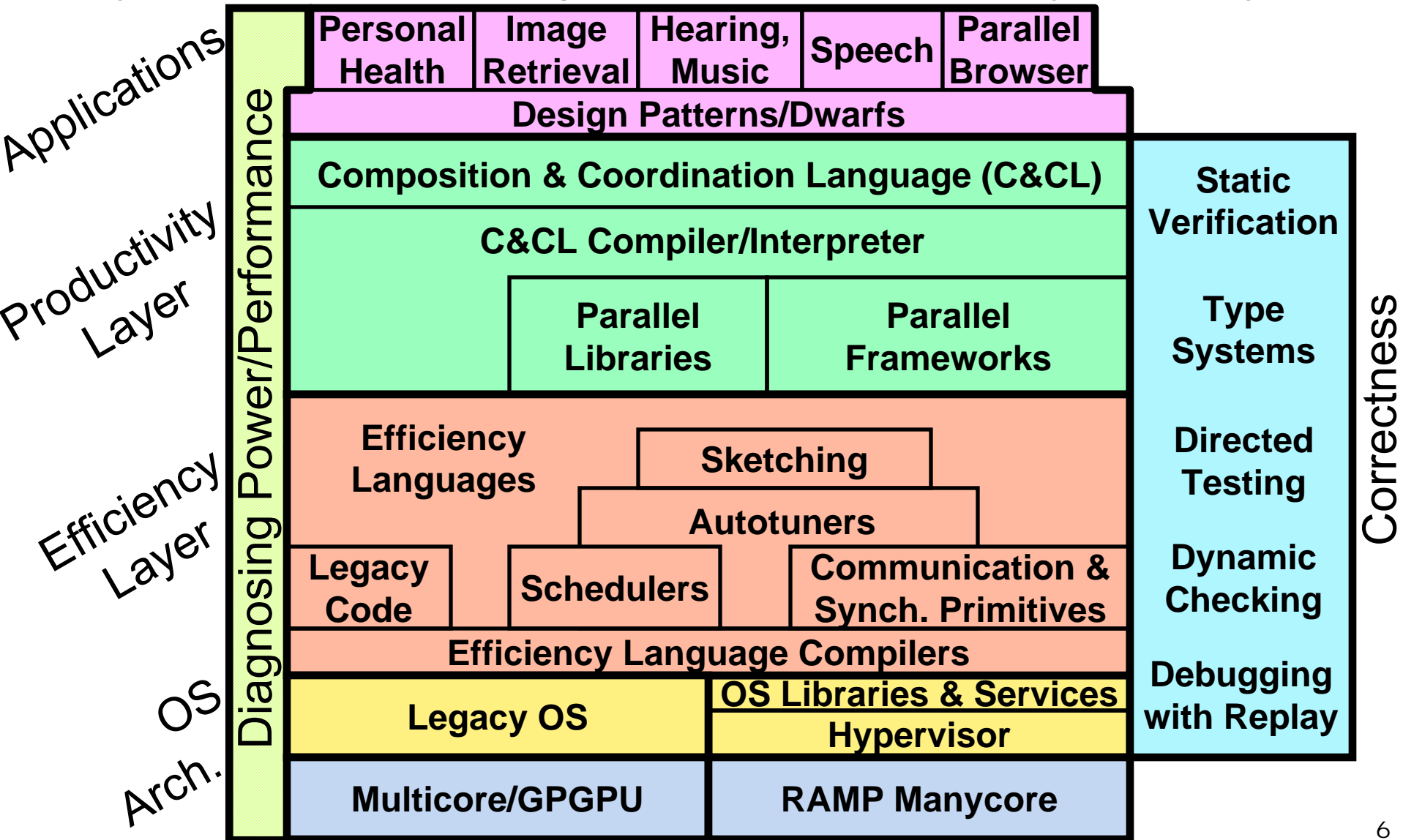
Deconstruction, Fast barrier synchronization, Partitions

### 5. Diagnosing Power/Performance Bottlenecks

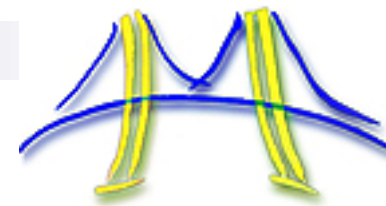
# Par Lab Research Overview



*Easy to write correct programs that run efficiently on manycore*



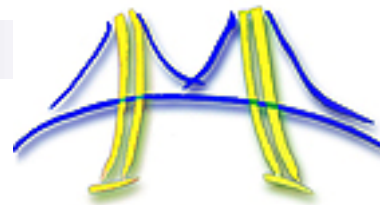
# What's the Big Idea?



- 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 3-4 yrs which ideas are big

# Compelling Laptop/Handheld Apps

## (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
  - Laptop/Handheld as accelerator for hearing aide
- 3. Novel Instrument User Interface
  - New composition and performance systems beyond keyboards
  - Input device for Laptop/Handheld

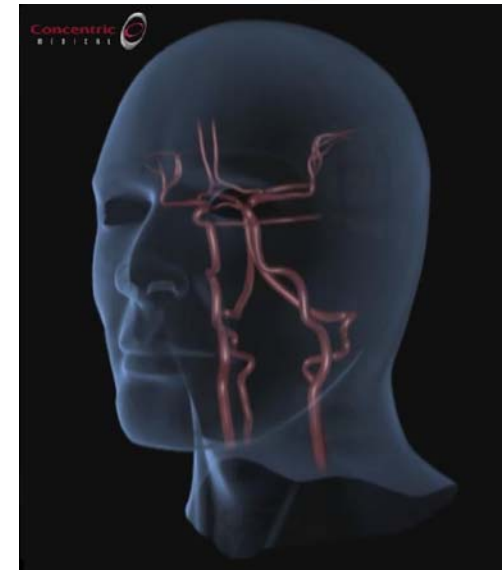
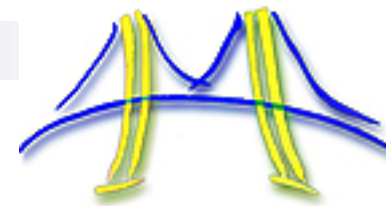


Berkeley Center for New Music and Audio Technology (CNMAT) created a compact loudspeaker array: 10-inch-diameter icosahedron incorporating 120 tweeters.

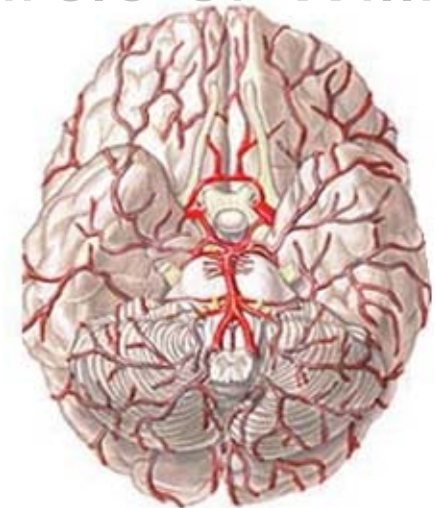
# Stroke diagnosis and treatment

(Tony Keaveny)

- 3<sup>rd</sup> deaths after heart, cancer
- No treatment >4 hours after
- Rapid Patient-specific 3D Fluid-Structure Interaction analysis of Circle of Willis
  - CoW 80% life-threatening strokes
  - Need highly-accurate simulations in near real-time
  - To evaluate treatment options while minimizing damage > 4 hrs after stroke



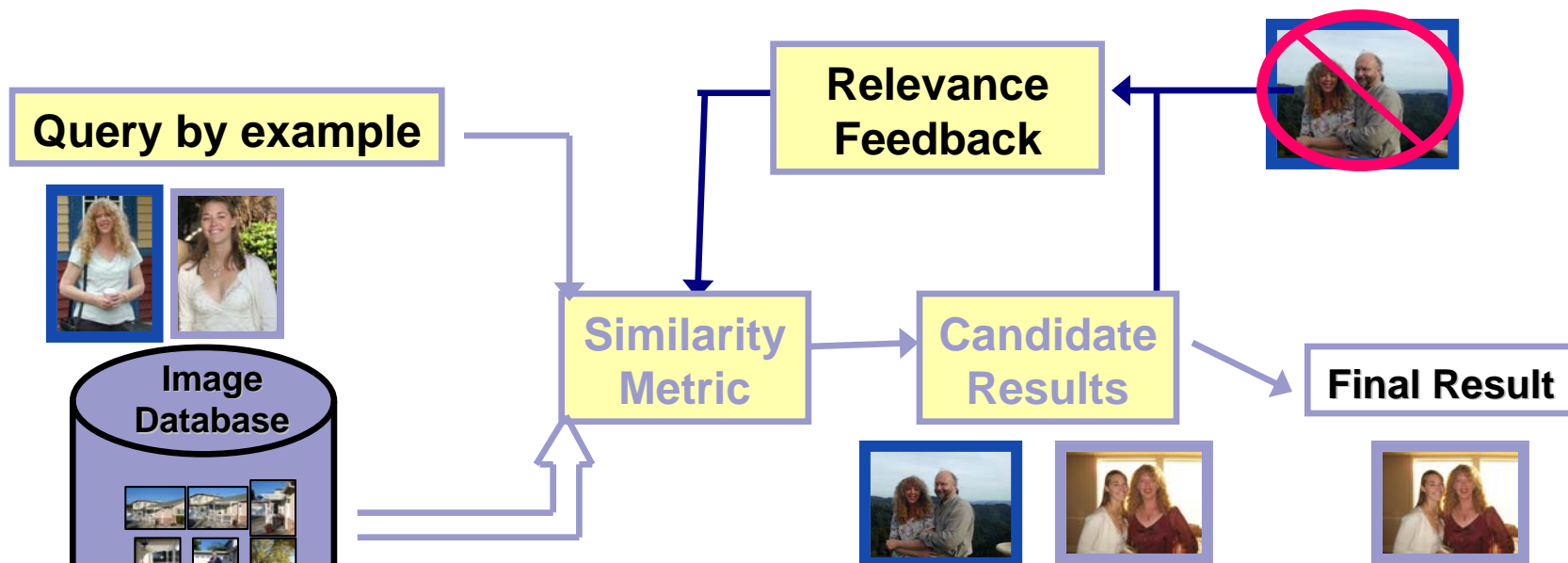
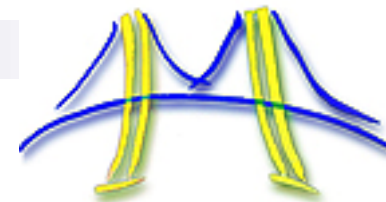
Circle of Willis



Bottom view of brain

# Content-Based Image Retrieval

(Kurt Keutzer)



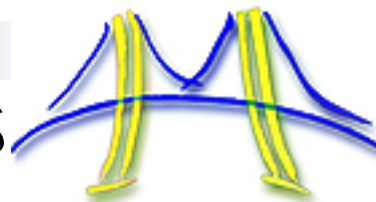
1000's of images

- 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

(Nelson Morgan)



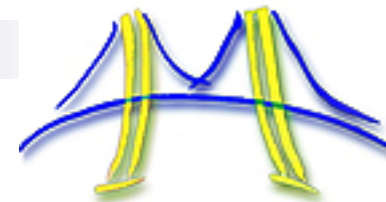
- Meeting Diarist
  - Laptops/ Handhelds at meeting coordinate to create speaker identified, partially transcribed text diary of meeting

The screenshot displays the AMI Meeting Browser interface. At the top, it shows the current slide 'Findings' with a list of materials: Materials for curved case (Plastic, Aluminum, Wood), Chips (Simple buttons, regular), and an advanced chip on print. Below this is a video feed of a meeting with participants Ed, Ayesha, Sudhir, and Christine. A central timeline shows speaker activity with colored bars (green, yellow, red) indicating who is speaking. On the right, an automatic transcript shows the text of the meeting, including phrases like 'she works in a cubicle next to missus she's that she was already a little bit prepared for the rest of the way yeah but it's not' and 'uh\_huh that you wouldn't have to have splinters in your hand point you are using'. At the bottom, a 'Dominance Levels' graph shows the relative speaking time of each participant over the course of the meeting, with a peak for Ed around 10:00. The AMI logo is visible in the bottom left corner.



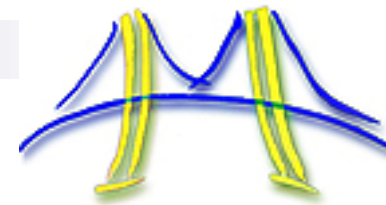
# Parallel Browser

## (Ras Bodik)

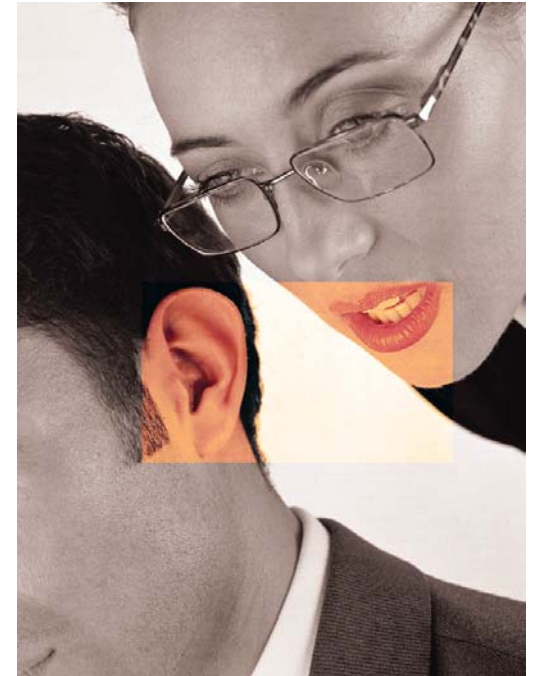


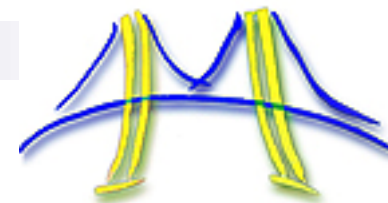
- Web 2.0: Browser plays role of traditional OS
  - Resource sharing and allocation, Protection
- Goal: Desktop quality browsing on handhelds
  - Enabled by 4G networks, better output devices
- Bottlenecks to parallelize
  - Parsing, Rendering, Scripting
- “SkipJax”
  - Parallel replacement for JavaScript/AJAX
  - Based on Brown’s FlapJax

# Compelling 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



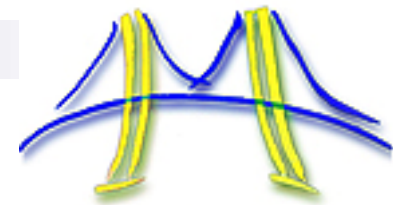


## Theme 2. What to compute?

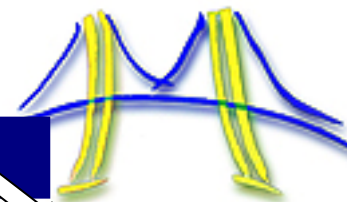
- Look for common computations across many areas
  1. Embedded Computing (42 EEMBC benchmarks)
  2. Desktop/Server Computing (28 SPEC2006)
  3. Data Base / Text Mining Software
  4. Games/Graphics/Vision
  5. Machine Learning / Artificial Intelligence
  6. Computer Aided Design
  7. High Performance Computing (Original "7 Dwarfs")
- Result: 12 Dwarfs

# "Dwarf" Popularity

(Red Hot \ Blue Cool)



- How do compelling apps relate to 12 dwarfs?



# Applications

<p>Choose your high level structure – what is the structure of my application? Guided expansion</p> <p>Pipe-and-filter Agent and Repository Process Control Event based, implicit invocation</p>	<p>Choose you high level architecture? Guided decomposition</p> <p>Task Decomposition ↔ Data Decomposition</p> <p>Group Tasks   Order groups   data sharing   data access</p> <p>Patterns?</p>	<p>Identify the key computation patterns – what are my key computations? Guided instantiation</p> <p>Graphical models Finite state machines Backtrack Branch and Bound N-Body methods Combinational Logic Spectral Methods</p>
<p>Model-view controller Bulk synchronous Map reduce Layered systems Arbitrary Static Task Graph</p>		<p>Graph Algorithms Dynamic Programming Dense Linear Algebra Sparse Linear Algebra Unstructured Grids Structured Grids</p>

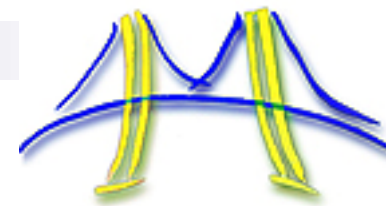
Productivity Layer

Refine the structure - what concurrent approach do I use? Guided re-organization				
Event Based Divide and Conquer	Data Parallelism Geometric Decomposition	Pipeline Discrete Event	Task Parallelism Graph Partitioning	Digital Circuits

Efficiency Layer

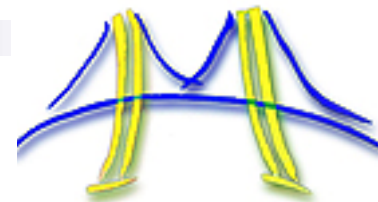
Utilize Supporting Structures – how do I implement my concurrency? Guided mapping				
Fork/Join CSP	Distributed Array Shared Data	Shared Queue Shared Hash Table	Master/worker Loop Parallelism	
Implementation methods – what are the building blocks of parallel programming? Guided implementation				
Thread Creation/destruction Process Creation/destruction	Message passing Collective communication	Speculation Transactional memory	Barriers Mutex	Semaphore s

# Themes 1 and 2 Summary

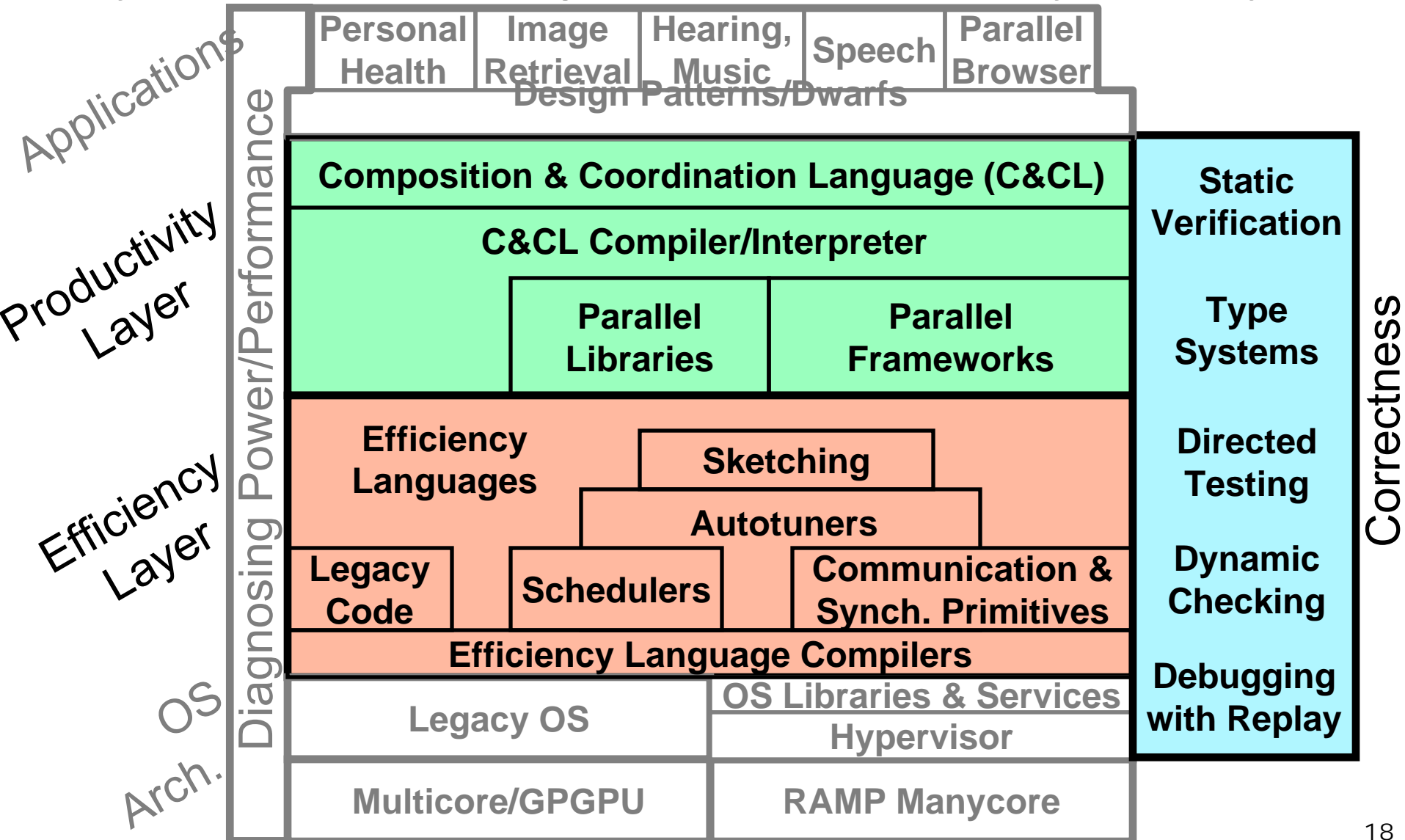


- Application-Driven Research (top down) vs. CS Solution-Driven Research (bottom up)
  - Bet is not that every program speeds up with more cores, but that we can find some compelling ones that do
- Drill down on (initially) 5 app areas to guide research agenda
- Dwarfs + Design Patterns to guide design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

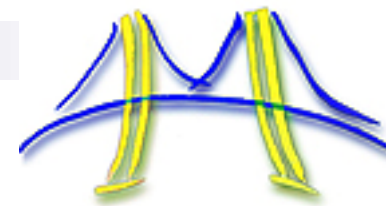
# Par Lab Research Overview



*Easy to write correct programs that run efficiently on manycore*



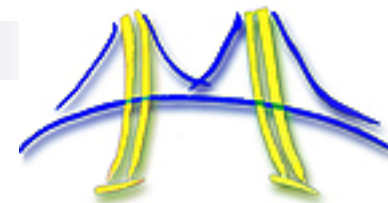
# Theme 3: Developing Parallel SW



- 2 types of programmers → 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; major challenge

# Ensuring Correctness

(Koushik Sen)



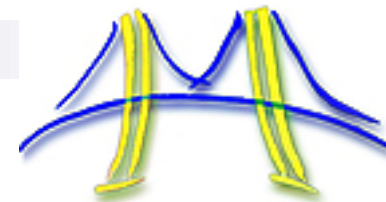
## ■ Productivity Layer

- 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)

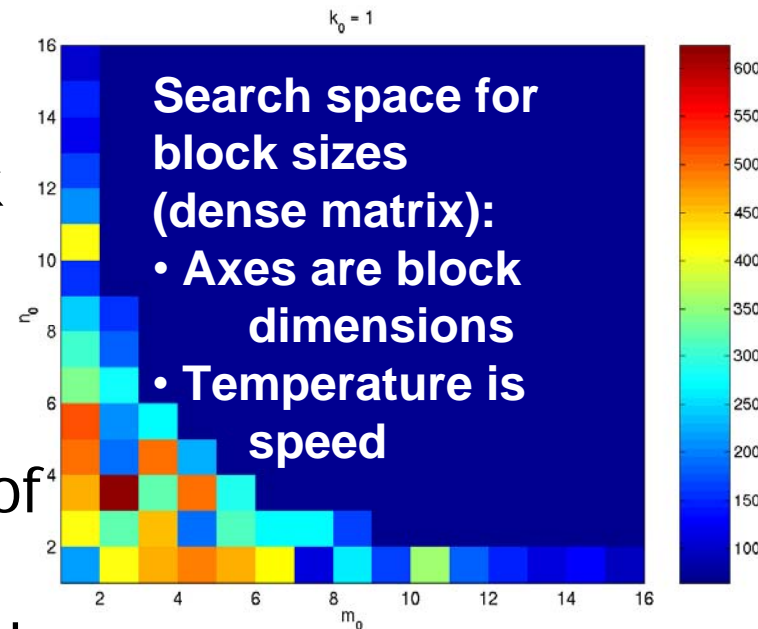
## ■ Efficiency Layer: Check for subtle concurrency bugs (races, deadlocks, and so on)

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

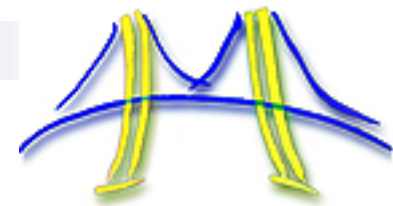
# 21<sup>st</sup> Century Code Generation (Demmel, Yelick)



- 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)



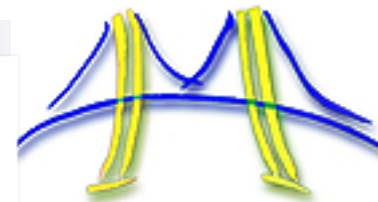
# Theme 3: Summary



- Autotuning vs. Static Compiling
- Productivity Layer & Efficiency Layer
- Composability of Libraries/Frameworks
- Libraries and Frameworks to leverage experts

# Par Lab

# Overview



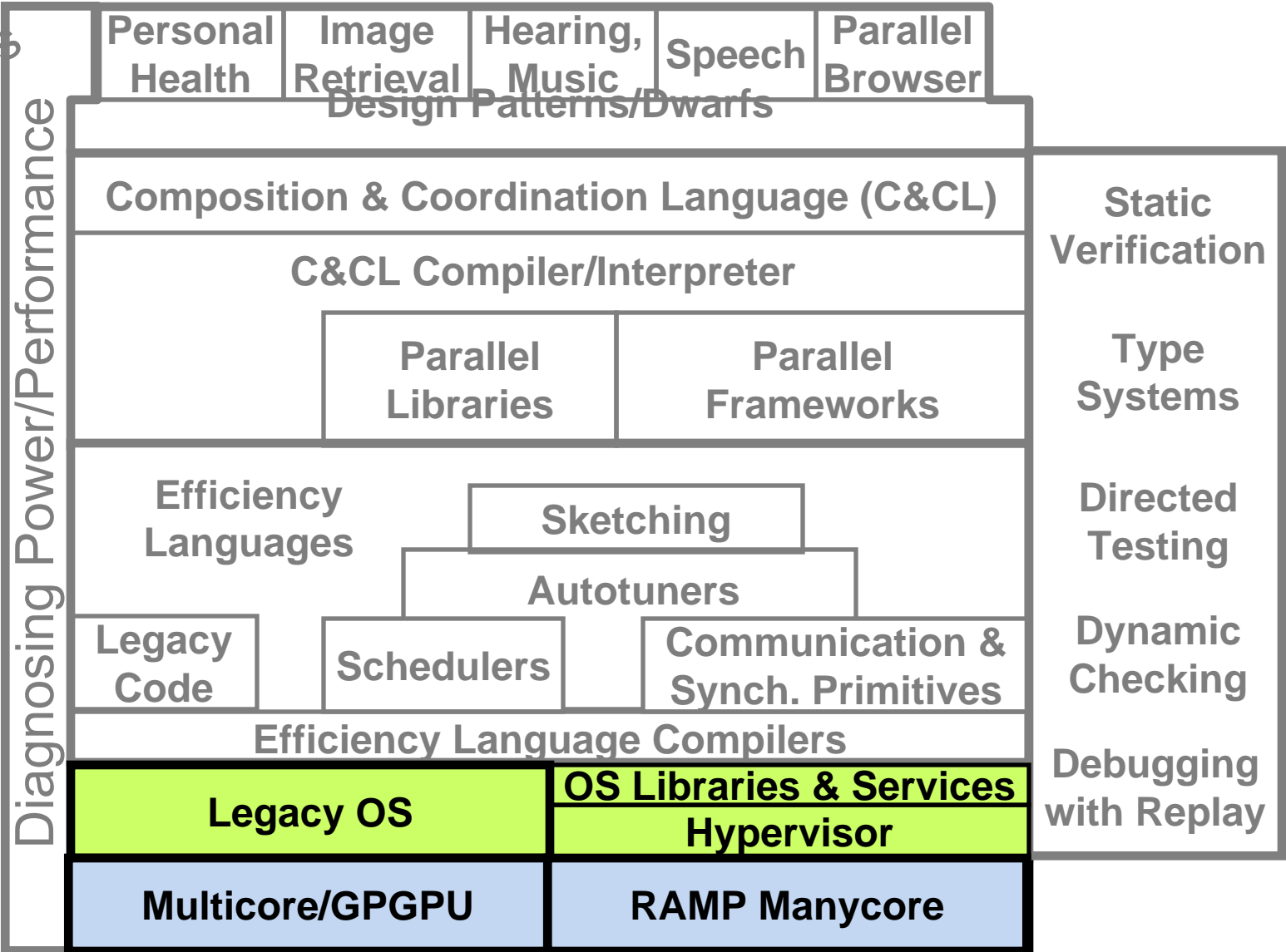
*Easy to write correct programs that run efficiently on manycore*

Applications

Productivity Layer

Efficiency Layer

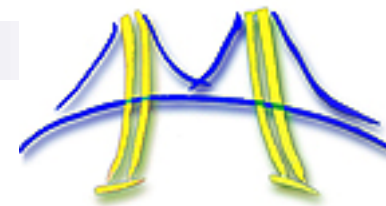
OS Arch.



Correctness

# Theme 4: OS and Architecture

(Krste Asanovic, Eric Brewer, John Kubiatawicz)



## ■ HW Solutions: Small is Beautiful

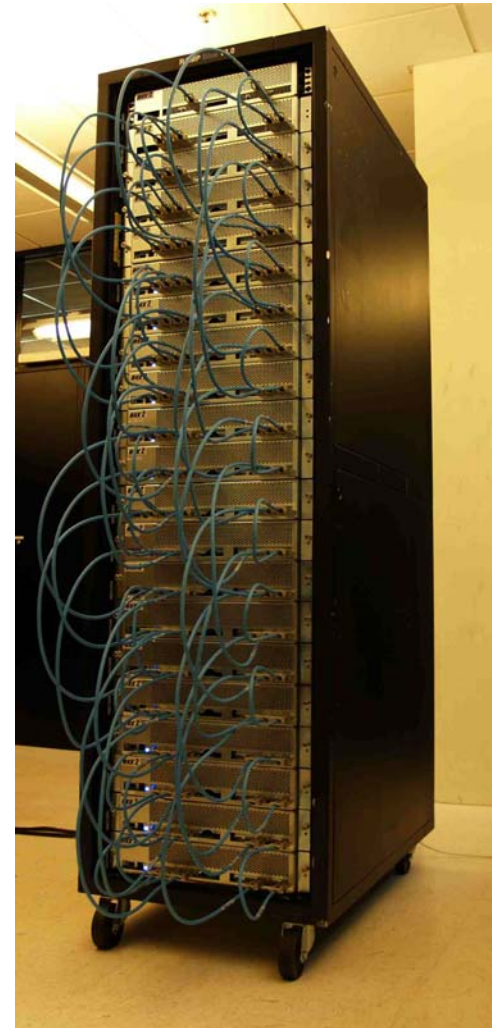
- Expect many modestly pipelined (5- to 9-stage) CPUs, FPUs, vector, SIMD Proc. Elmts
- Reconfigurable Memory Hierarchy
- Offer HW partitions with 1-ns Barriers

## ■ Deconstructing Operating Systems

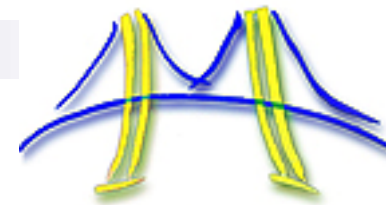
- Resurgence of interest in virtual machines
- Leverage HW partitioning for thin hypervisors
  - ➔ Allow SW full access to HW in partition

## 1008 Core "RAMP Blue" (Wawrzynek, Asanovic)

- 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 manufacturing and selling boards
  - Gateware, Software BSD open source

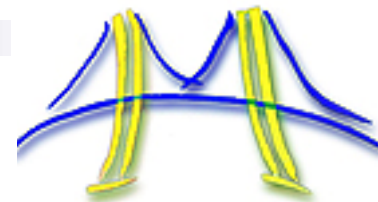


# Par Lab Domain Expert Deal

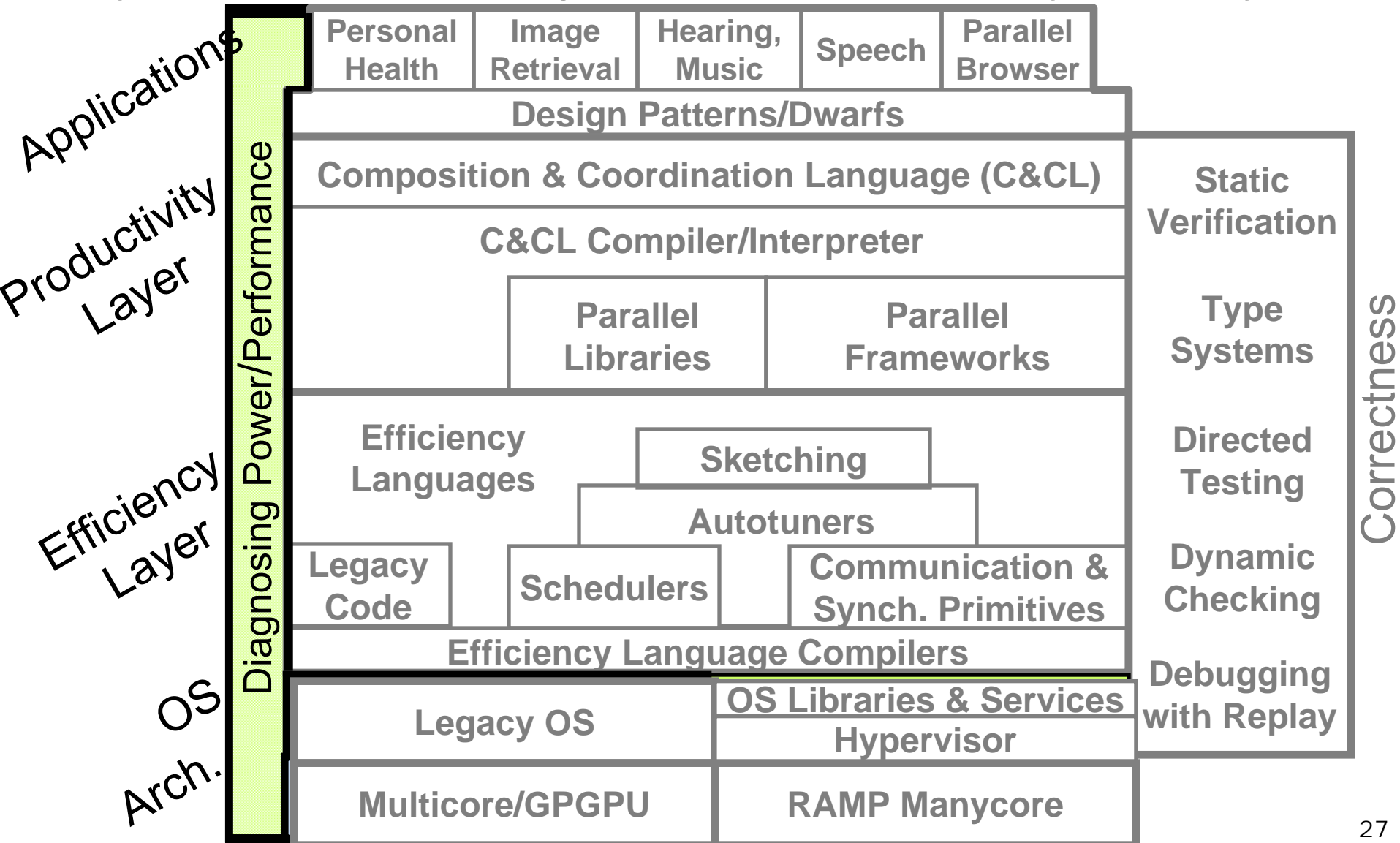


- Get help developing application on latest commercial multicores / GPUs and legacy OS
  - + Develop using many fast, recent, stable computers
  - + Develop on preproduction version of new computers
  - Conventional architectures and OS, but many types
- Will help port app to innovative Par Lab Arch and OS implemented in “RAMP Gold”
  - + Arch & OS folk innovate for (your) app of future (vs. benchmarks of past)
  - + Use computer with as many cores as you want and world’s best measurement, diagnosis, & debug HW
  - Runs 20X slower than commercial hardware

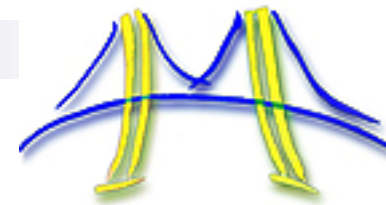
# Par Lab Research Overview



*Easy to write correct programs that run efficiently on manycore*

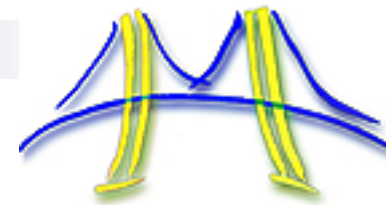


# Theme 5: Diagnosing Power/ Performance Bottlenecks (Demmel)

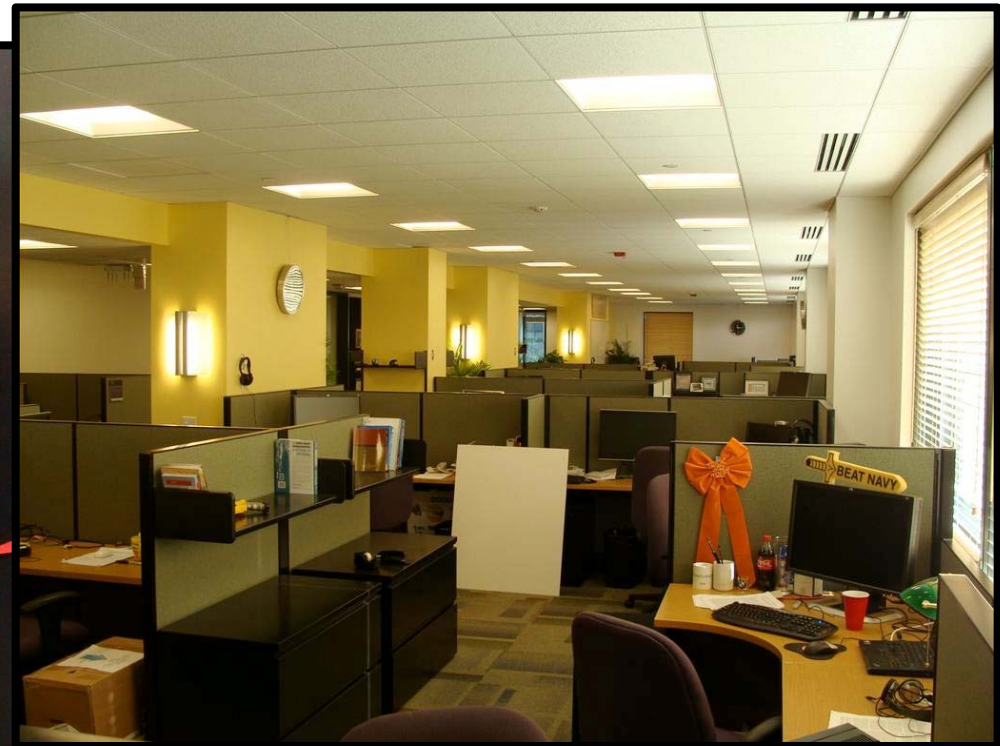


- Collect data on Power/Performance bottlenecks
  - Aid autotuner, scheduler, OS in adapting system
- Turn into info to help *efficiency*-level programmer?
  - Am I using 100% of memory bandwidth?
- Turn into info to help *productivity* programmer?
  - If I change it like this, impact on Power/Performance?
- An IEEE Counter Standard for all multicores?
  - => Portable performance tool kit, OS scheduling aid
  - Measuring utilization accurately >> New Optimization
    - If saves 20% performance, why not worth 10% resources?
  - RAMP Gold 1st implementation, help evolve standard

# New Par Lab: Opened Dec 1, 2008

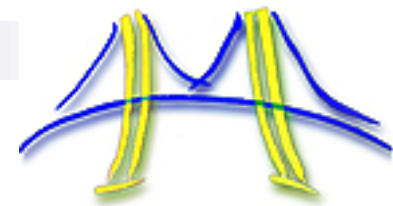


- 5<sup>th</sup> Floor South Soda Hall South (565 Soda)
- Founding Partners: Intel and Microsoft
- 1st Affiliate Partners: Samsung and NEC

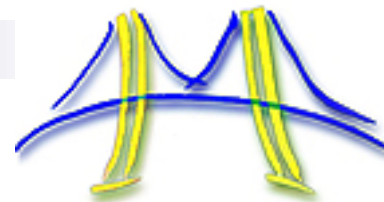


# Recent Results: Active Testing

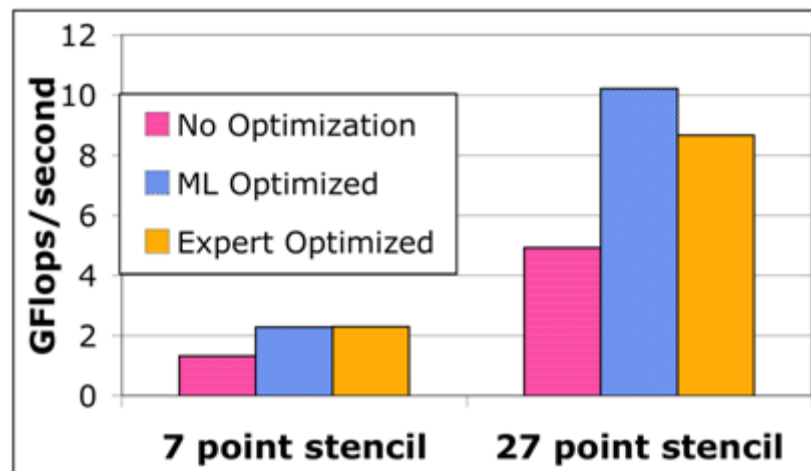
- Pallavi Joshi and Chang-Seo Park
- Problem: Concurrency Bugs
- Actively control the scheduler to force potentially buggy schedules: Data races, Atomicity Violations, Deadlocks
- Found parallel bugs in real OSS code: Apache Commons Collections, Java Collections Framework, Jigsaw web server, Java Swing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



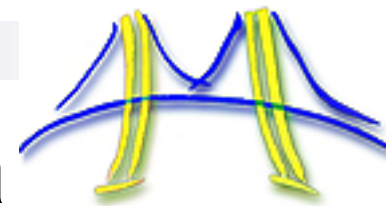
# Results: Making Autotuning “Auto”



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



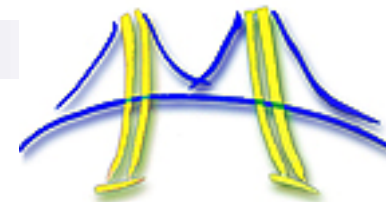
# Results: Fast Dense Linear Algebra



- Mark Hoemmen: LINPACK benchmark made dense linear algebra seem easy
  - If solve impractically large problems ( $10^6 \times 10^6$ )
- Problem: Communication limits perf. for non-huge matrices and increasing core counts
- New way to panel matrix to minimize comm.
  - "Tall Skinny" QR factorization
- IBM BlueGene/L, 32 cores: up to 4× faster
- Pentium III cluster, 16 cores: up to 6.7× faster
  - vs. Parallel LINPACK (ScaLAPACK) on  $10^5 \times 200$  matrix

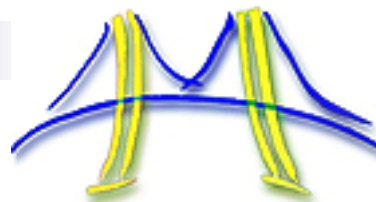


# Recent Results: App Acceleration



- Bryan Catanzaro: Parallelizing Computer Vision (image segmentation) using GPU
- Problem: On PC Malik's highest quality algorithm is 7.8 minutes / image
- 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: 2.5 seconds / image
- ~ 200X speedup
  - Factor of 10 quantitative change is a qualitative change
- Malik: "This will revolutionize computer vision."

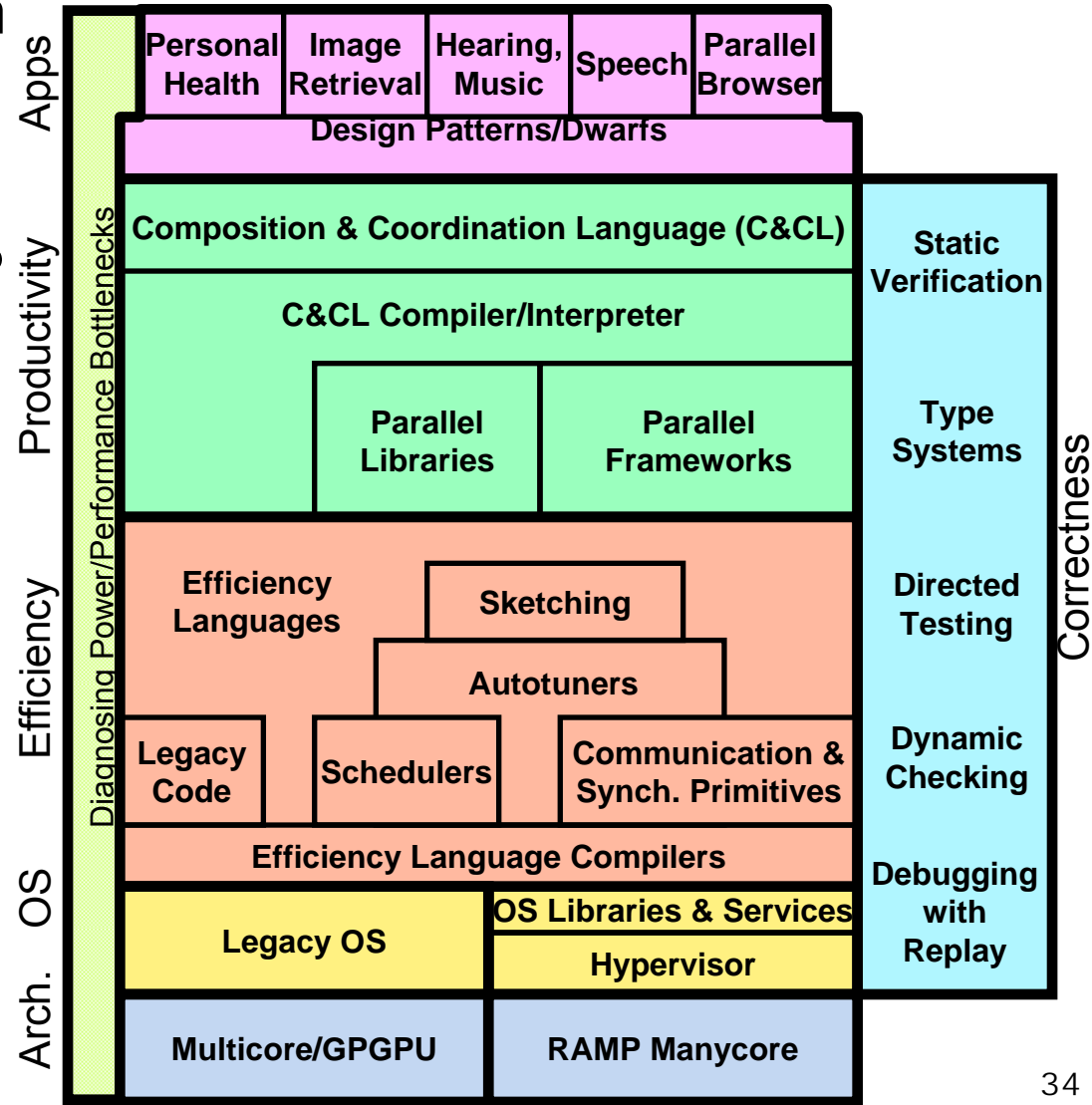




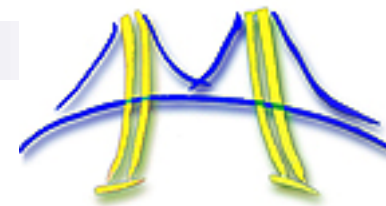
# Par Lab Summary

- Try Apps-Driven vs. CS Solution-Driven Research
- Design patterns + Dwarfs
- Efficiency layer for  $\approx 10\%$  today's programmers
- Productivity layer for  $\approx 90\%$  today's programmers
- Autotuners vs. Compilers
- OS & HW: Primitives vs. Solutions
- Verification Directed Testing
- Counter Standard to find Power/Perf. bottlenecks

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



# Acknowledgments



- Faculty, Students, and Staff in Par Lab
- Intel and Microsoft for being founding sponsors of Par Lab; Samsung and NEC as 1<sup>st</sup> Affiliate Members
- Contact me if interested in becoming Par Lab Affiliate (pattrsn@cs.berkeley.edu)
- **See [parlab.eecs.berkeley.edu](http://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](http://ramp.eecs.berkeley.edu)**