Microsoft® Research

Faculty Summit

10

YEAR ANNIVERSARY

Microsoft Research

UC Berkeley Par Lab Overview

David Patterson

Par Lab's original research "bets"

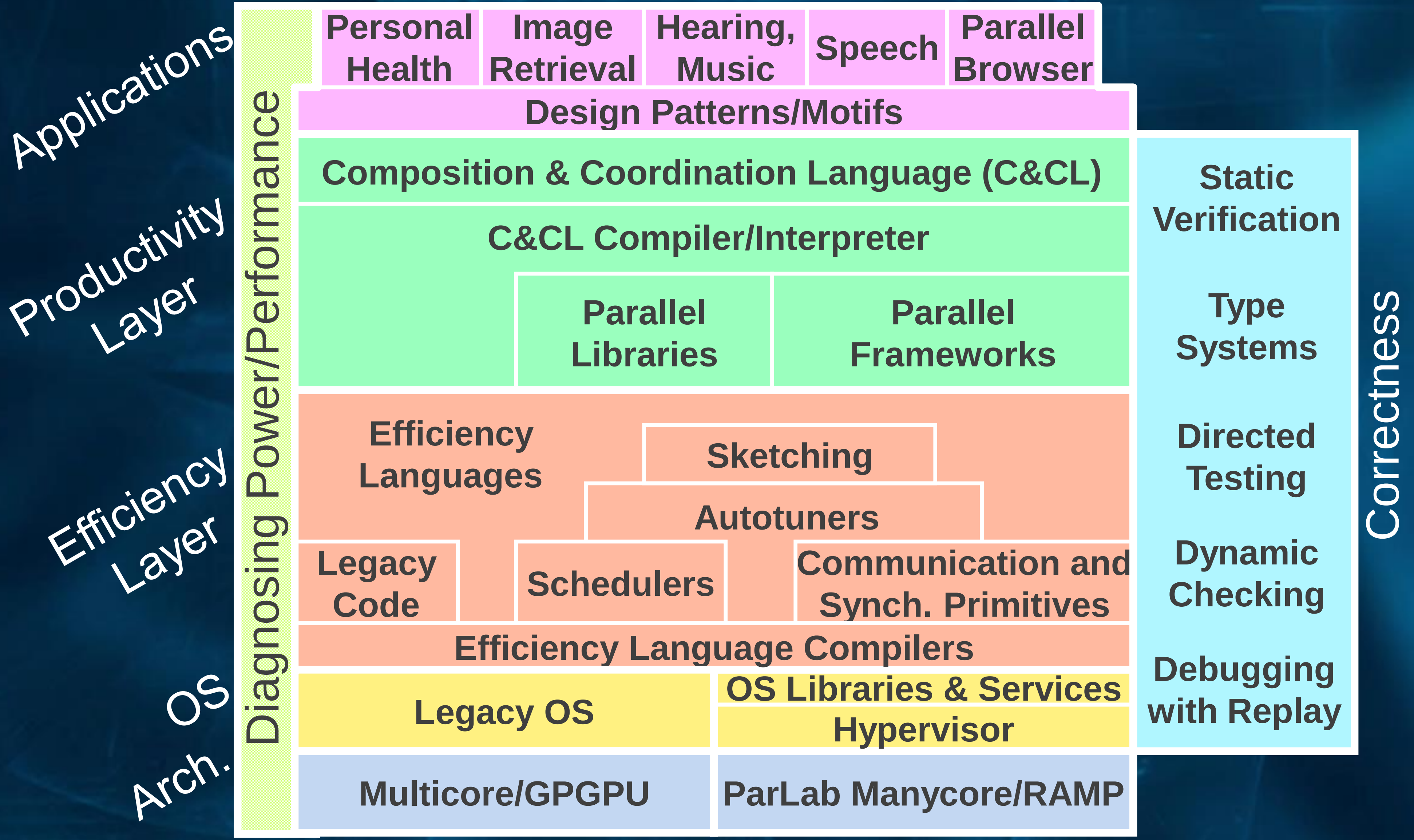
Microsoft Research

- Software platform: data center + mobile client
- Let compelling applications drive research agenda
- 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

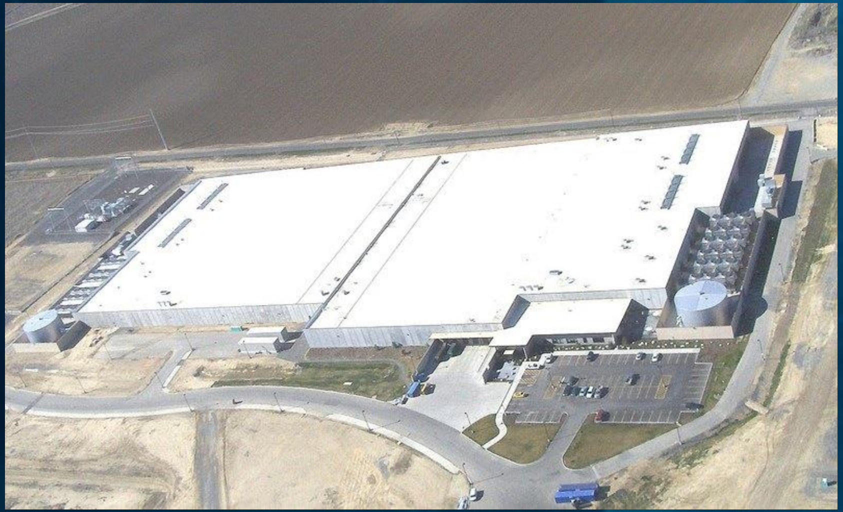
Above all, no preconceived big idea –
see what works driven by application needs

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)

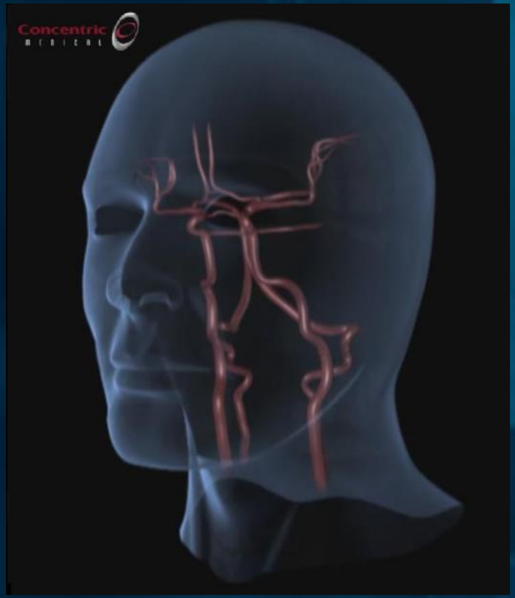
Microsoft Research

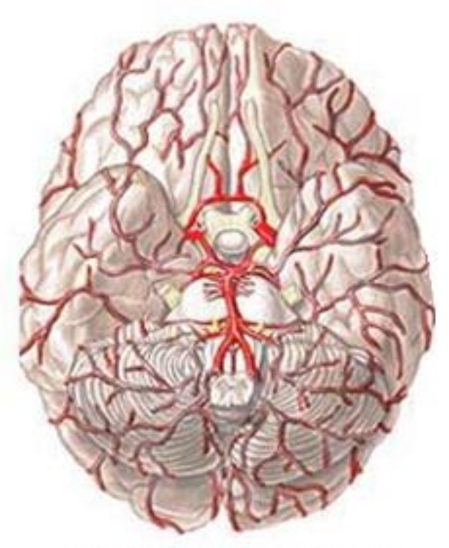
- Musicians have an insatiable appetite for computation + real-time demands
  - More channels, instruments, more processing, more interaction!
  - Latency must be low (5 ms)
  - Must be reliable (No clicks!)

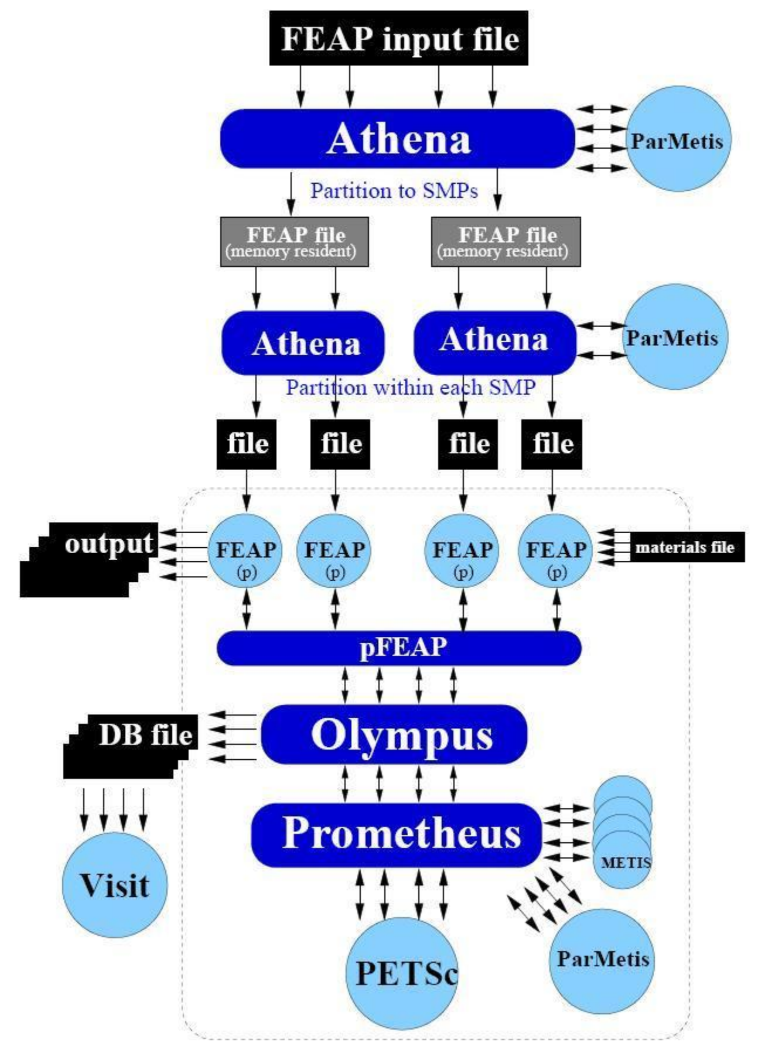
1. Music Enhancer
   - Enhanced sound delivery systems for home sound systems using large microphone and speaker arrays
   - Laptop/Handheld recreate 3D sound over ear buds
2. Hearing Augmenter
   - Handheld as accelerator for hearing aid
3. Novel Instrument User Interface
   - New composition and performance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




Bottom view of brain
© ADAM, Inc.

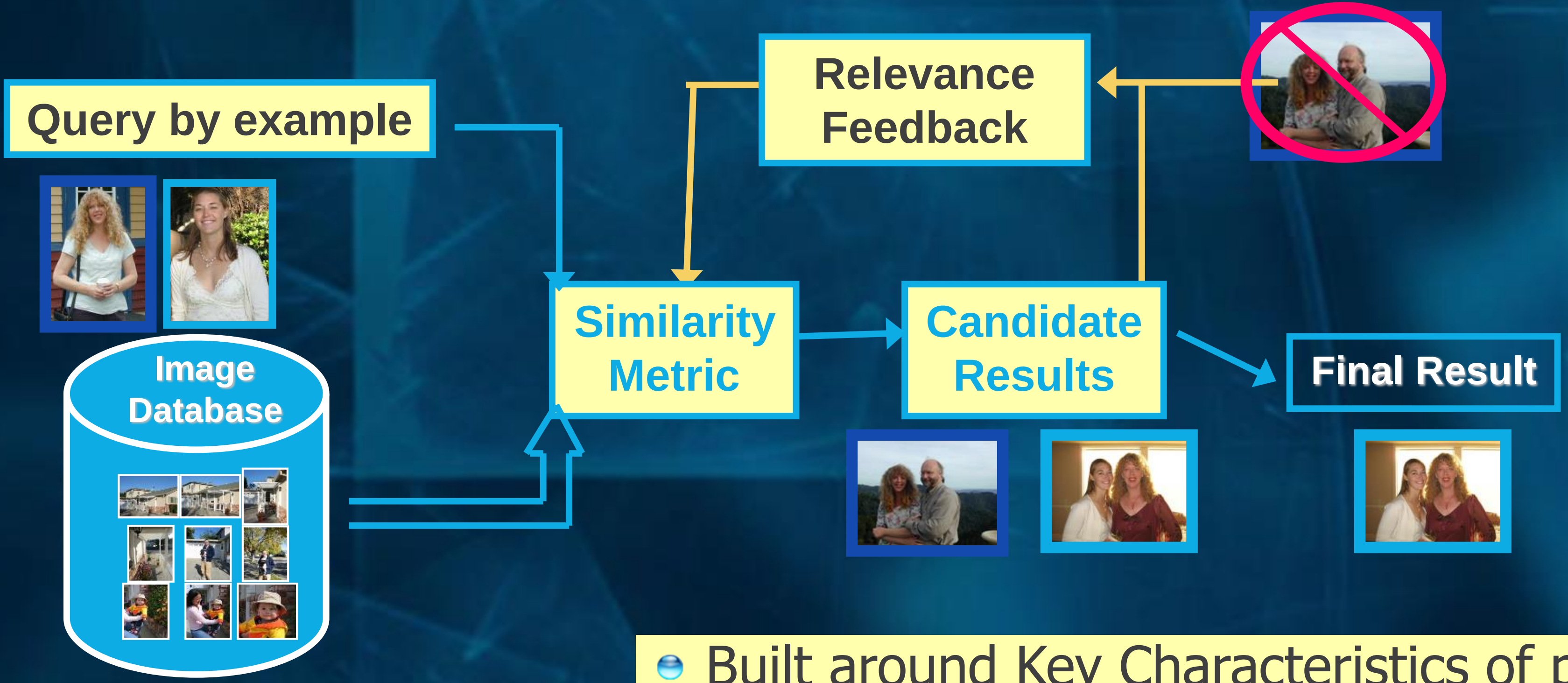


- Stroke treatment time-critical, need supercomputer performance in hospital
- Goal: First true 3D Fluid-Solid Interaction analysis of Circle of Willis
- Based on existing codes for distributed clusters

Content-Based Image Retrieval

(Kurt Keutzer)

Microsoft Research



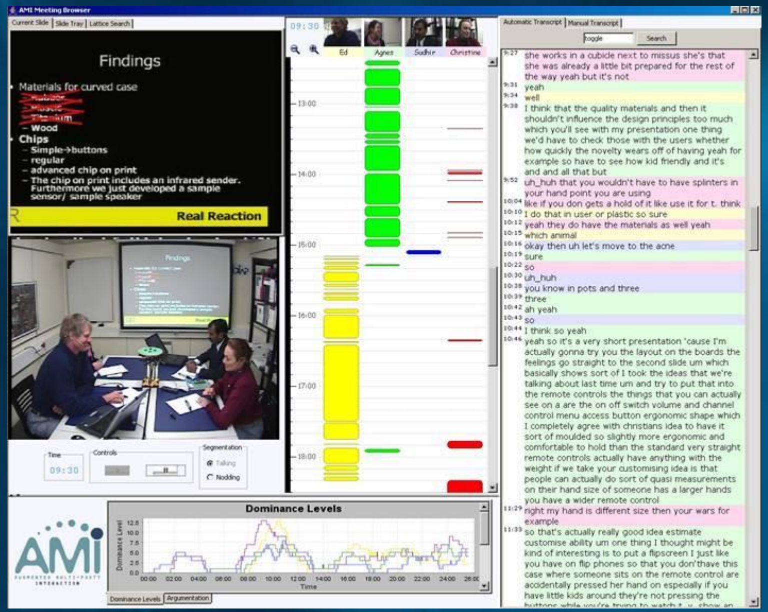
1000's of images

- 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

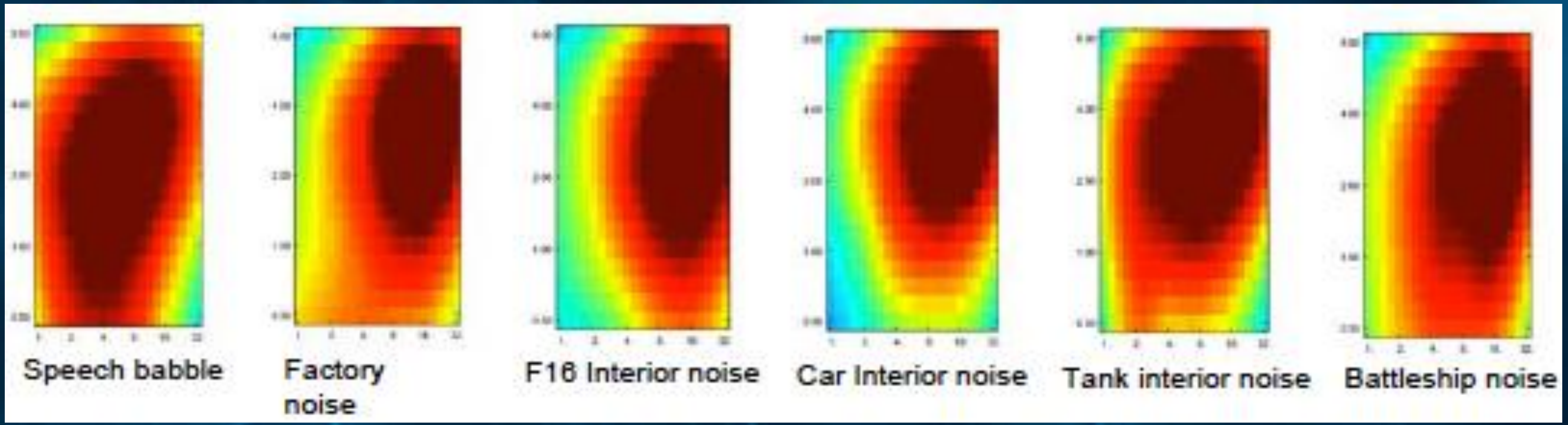
Robust Speech Recognition

(Nelson Morgan)

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

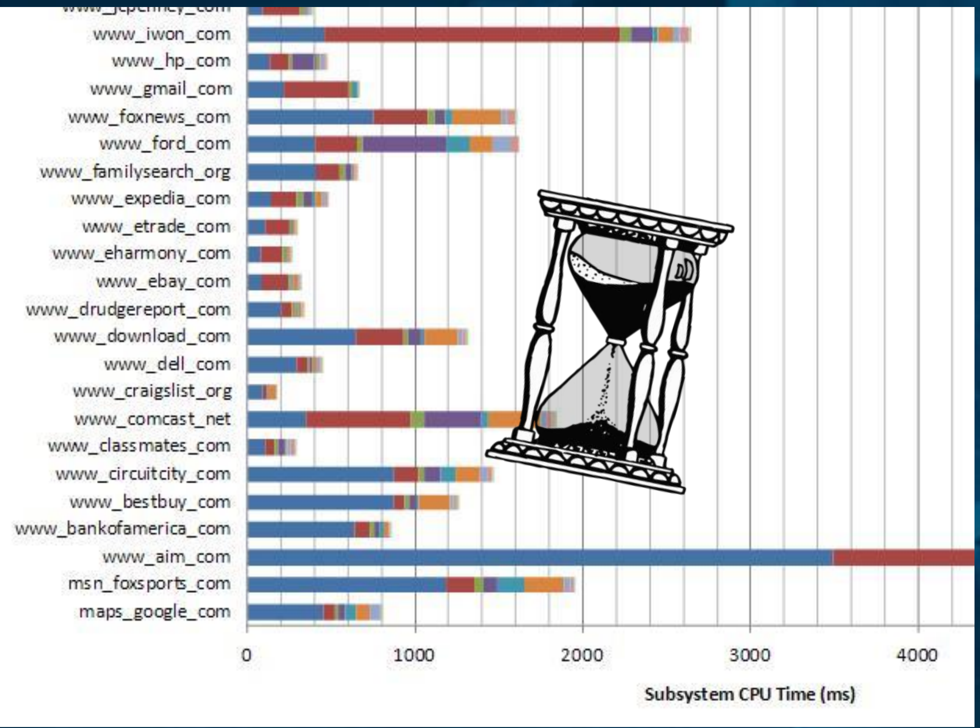


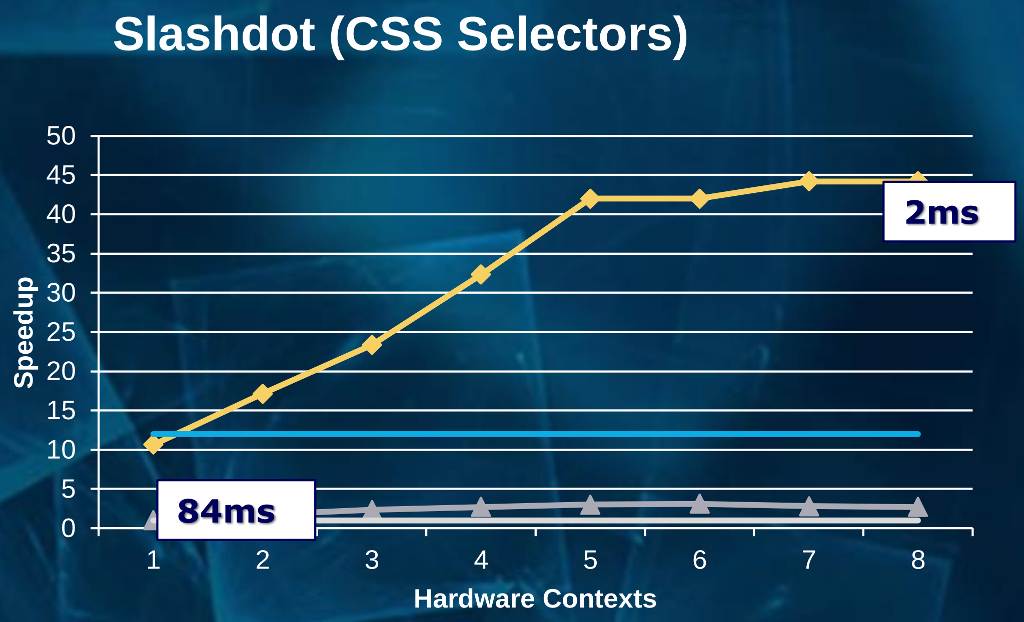
- Use cortically-inspired manystream spatio-temporal features to tolerate noise

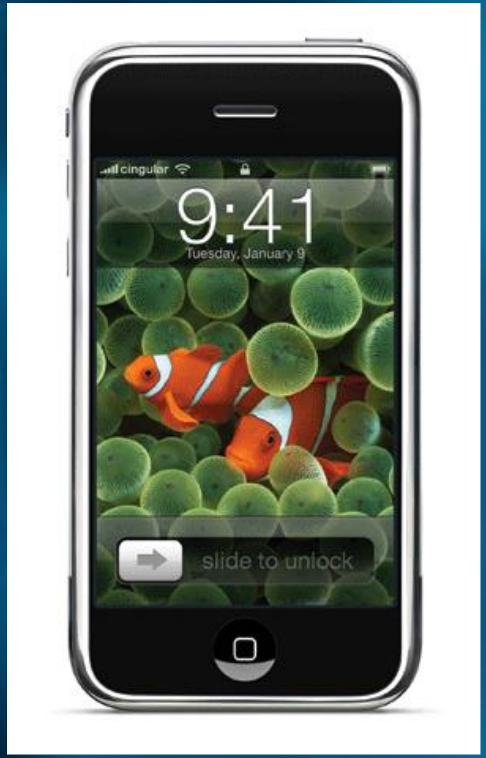


Parallel Browser
(Ras Bodik)

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



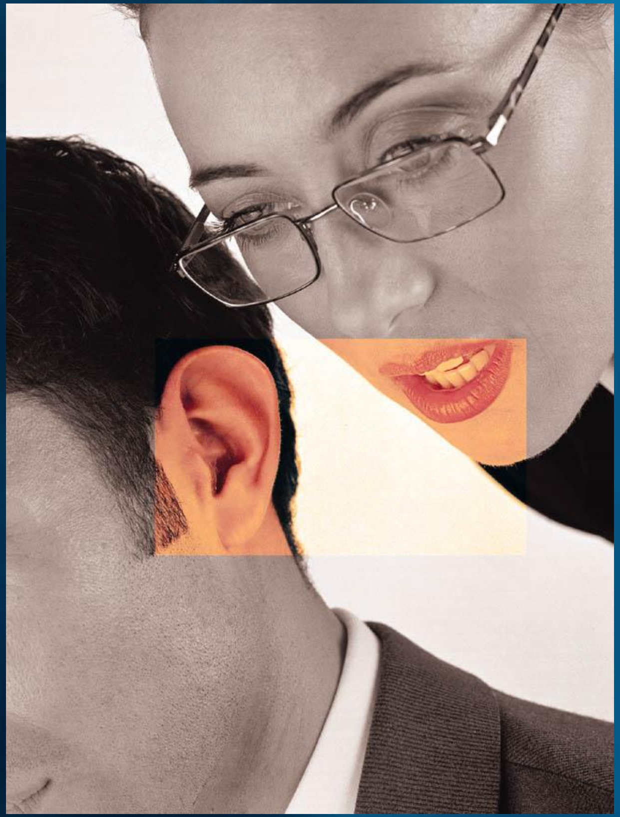




Microsoft Research

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)

Microsoft Research

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

"Dwarfs" -> Motifs

- Computational patterns
- Structural patterns

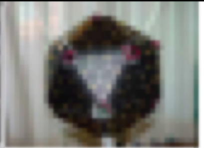
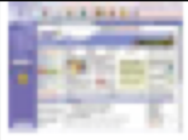
Insight: Successful codes have a comprehensible software architecture:

- Patterns give human language in which to describe architecture

Motif (nee "Dwarf") Popularity
(Red Hot \ Blue Cool)

Microsoft Research

- How do compelling apps relate to 12 motifs?

		Embed	SPEC	DB	Games	ML	CAD	HPC	 Health	 Image	 Speech	 Music	 Browser
1	Finite State Mach.	Red	Red	Red	Yellow	Yellow	Yellow	Blue	Blue	Blue	Blue	Blue	Red
2	Circuits	Red	Blue	Green	Blue	Green	Blue	Blue	Blue	Blue	Blue	Blue	Red
3	Graph Algorithms	Red	Yellow	Yellow	Yellow	Red	Red	Blue	Red	Blue	Red	Green	Green
4	Structured Grid	Red	Red	Blue	Yellow	Blue	Blue	Red	Blue	Red	Blue	Blue	Blue
5	Dense Matrix	Red	Red	Yellow	Red	Red	Red	Red	Blue	Red	Red	Red	Blue
6	Sparse Matrix	Yellow	Yellow	Blue	Red	Red	Red	Red	Red	Blue	Blue	Red	Blue
7	Spectral (FFT)	Yellow	Blue	Blue	Yellow	Yellow	Yellow	Red	Blue	Green	Red	Red	Red
8	Dynamic Prog	Yellow	Blue	Red	Blue	Red	Red	Blue	Blue	Blue	Yellow	Blue	Red
9	Particle Methods	Blue	Yellow	Blue	Yellow	Blue	Blue	Red	Blue	Blue	Blue	Blue	Blue
10	Backtrack/ B&B	Blue	Blue	Yellow	Blue	Red	Red	Blue	Blue	Blue	Blue	Yellow	Blue
11	Graphical Models	Blue	Blue	Yellow	Blue	Red	Blue	Blue	Blue	Blue	Blue	Red	Blue
12	Unstructured Grid	Blue	Blue	Blue	Yellow	Yellow	Yellow	Red	Red	Blue	Blue	Red	Blue

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

People, Patterns, and Frameworks

Microsoft Research

	  Design Patterns	  Frameworks
Application Developer 	Uses application design patterns (e.g. feature extraction) to architect the application	Uses application frameworks (e.g. CBIR) to implement the application
Application-Framework Developer 	Uses programming design patterns (e.g. Map/Reduce) to architect the application framework	Uses programming frameworks (e.g MapReduce) to implement the application framework

Productivity/Efficiency and Patterns

Microsoft Research

1. Domain Experts + Application patterns and frameworks → End-user, application programs

2. Domain-literate programming gurus (1% of the population) + Parallel patterns and programming frameworks → Application frameworks

3. Parallel programming gurus (1-10% of programmers) → Parallel programming frameworks

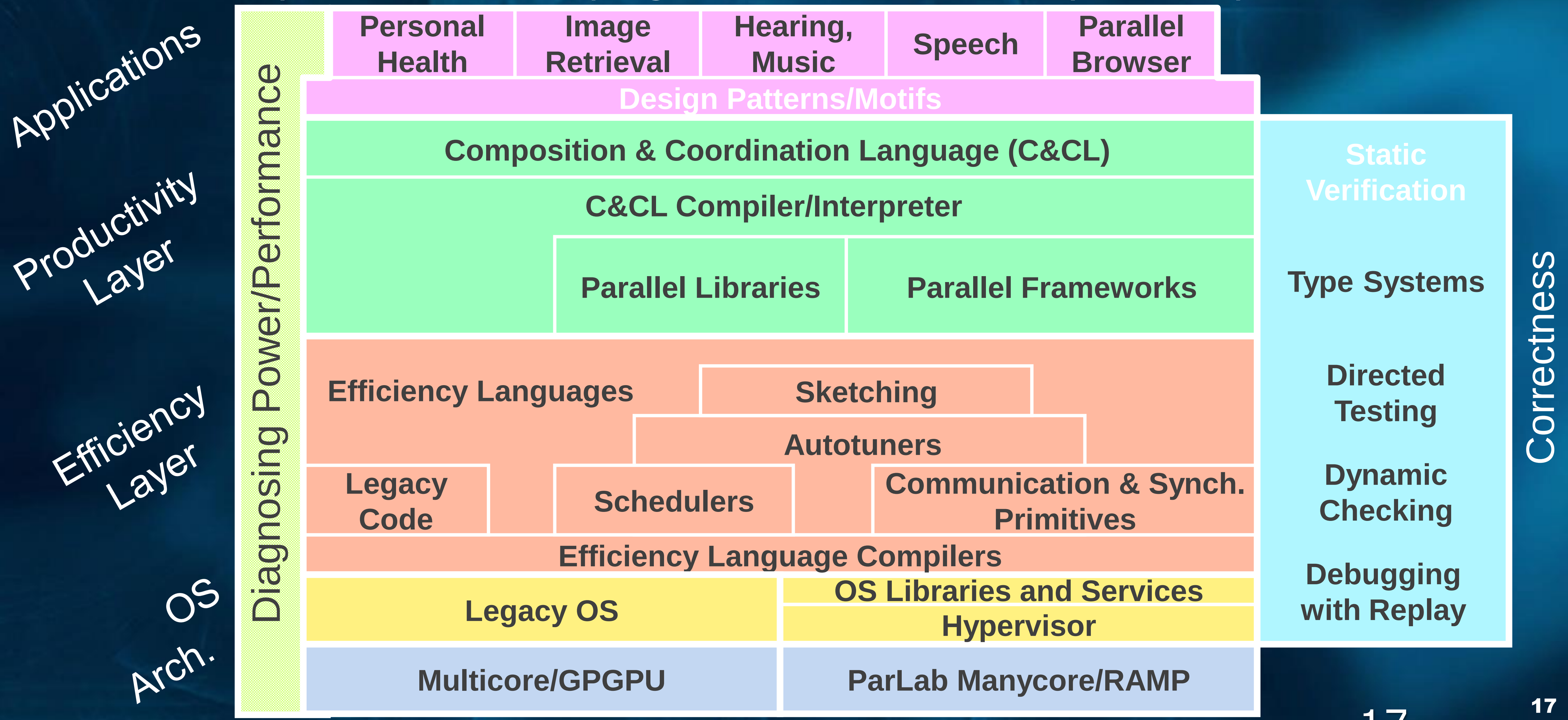
The hope is for Domain Experts to create parallel code with little or no understanding of parallel programming

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

Par Lab Research Overview

Microsoft Research

Easy to write correct programs that run efficiently on manycore



Par Lab is Multi-Lingual

Microsoft Research

- 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

Why Consider New Languages?

Microsoft Research

- Most of work is in runtime and libraries
- Do we need a language? And a compiler?
  - If higher level syntax is needed for productivity
    - We need a language
  - If static analysis is needed to help with correctness
    - We need a compiler (front-end)
  - If static optimizations are needed to get performance
    - We need a compiler (back-end)
- Will prototype frameworks in conventional languages, but investigate how new languages or pattern-specific compilers can improve productivity, efficiency, and/or correctness

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

Microsoft Research

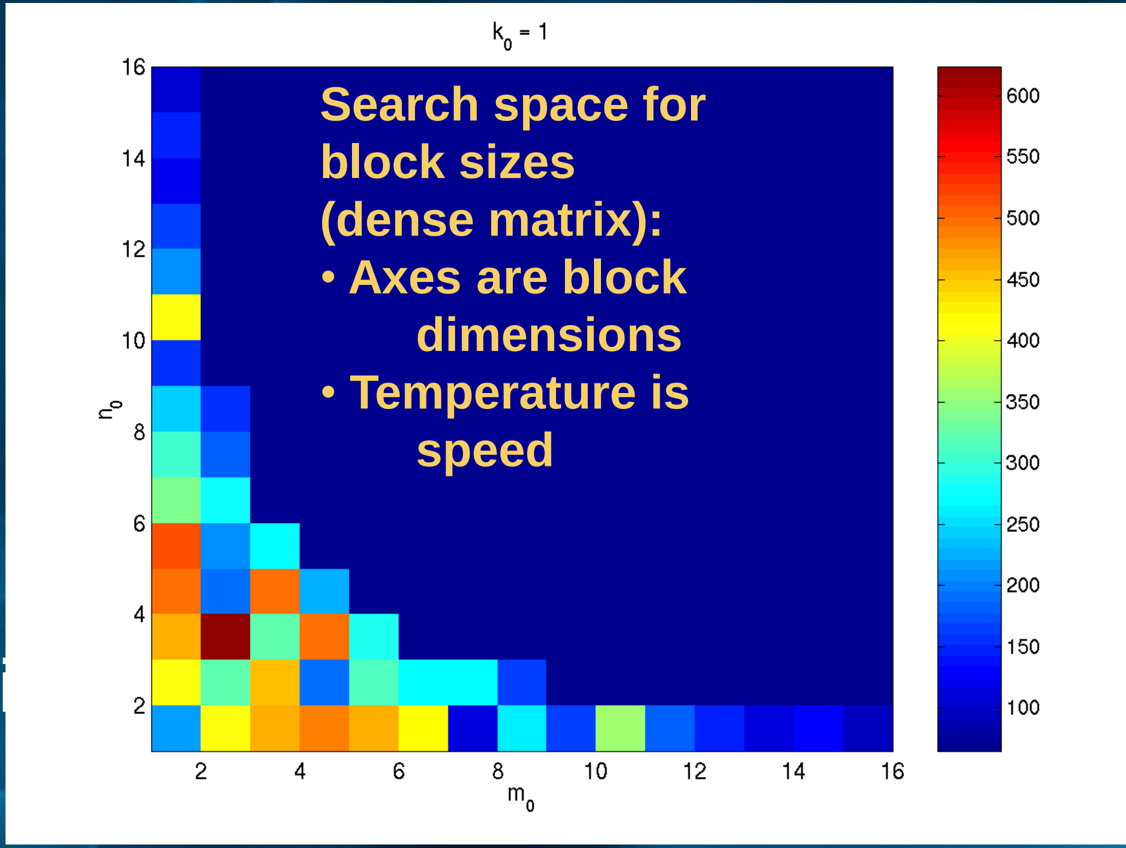
- Case Study: Stencil Kernels on AMD Barcelona, 8 threads
- Hand-coded in C/OpenMP: 2-4 days
- SEJITS in Ruby: 1-2 hours
- Time to run 3 stencil codes:

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

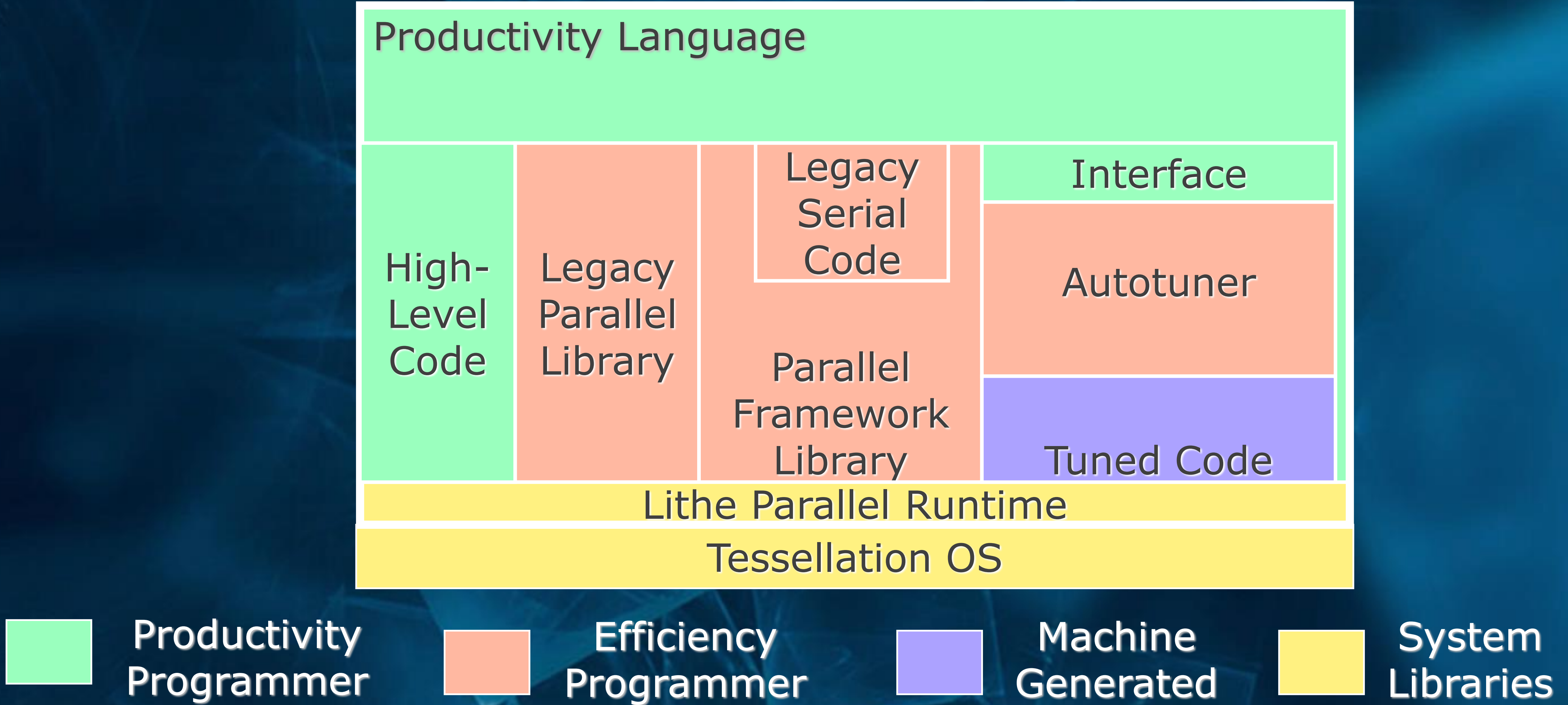
Autotuning for Code Generation (Demmel, Yelick)

Microsoft Research

- 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 (blocki 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)



Anatomy of a Par Lab Application

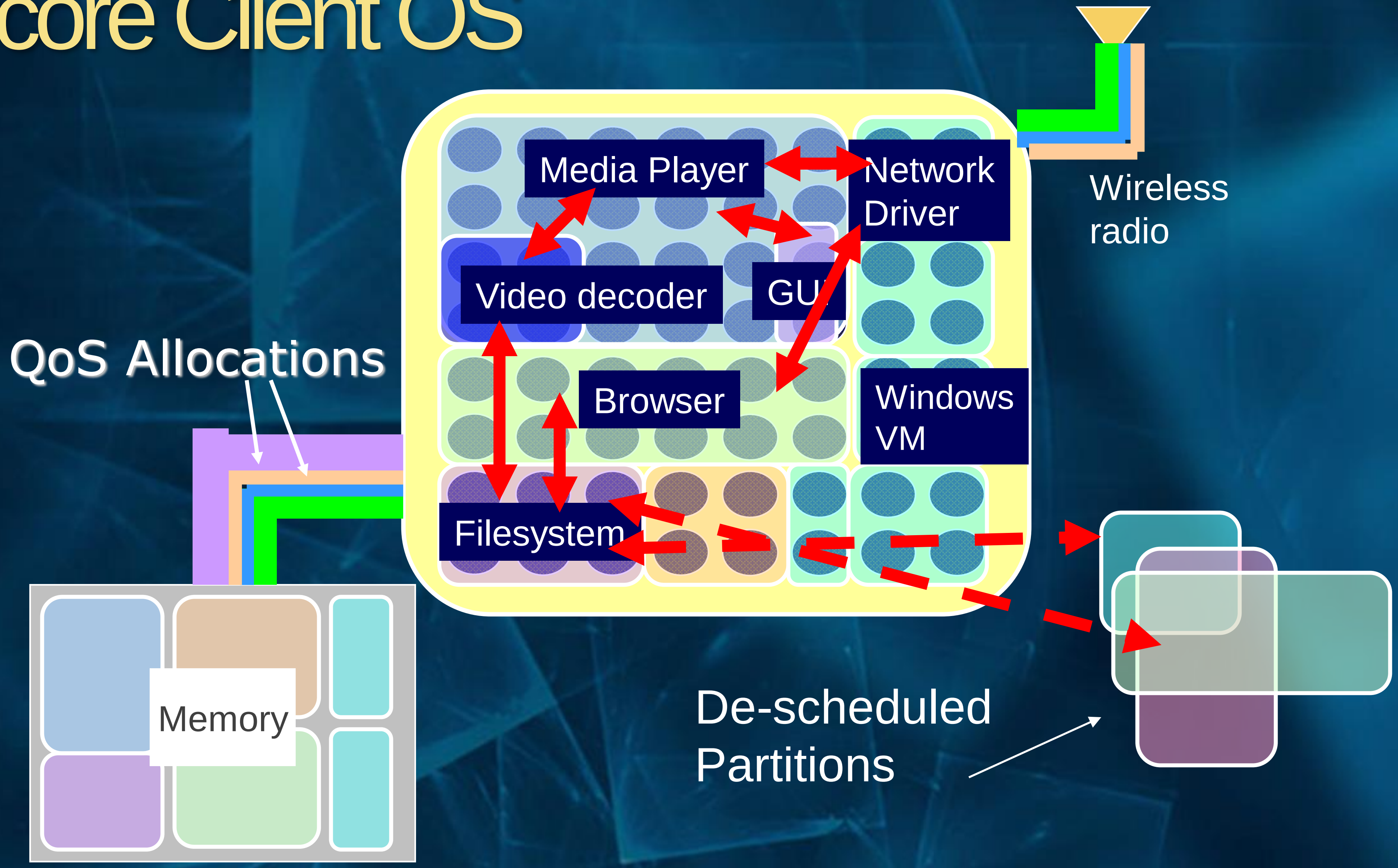


From OS to User-Level Scheduling

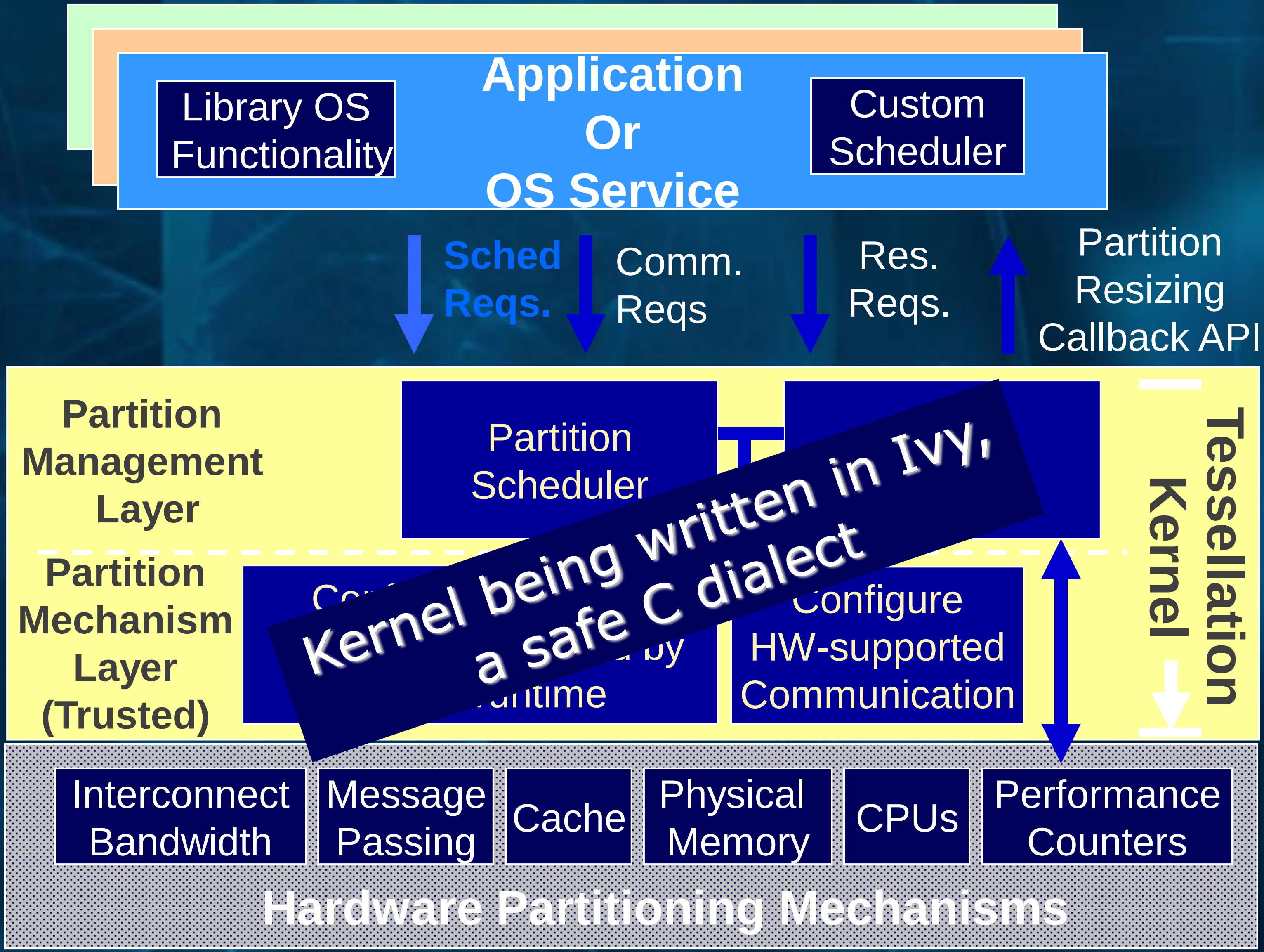
Microsoft Research

- 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



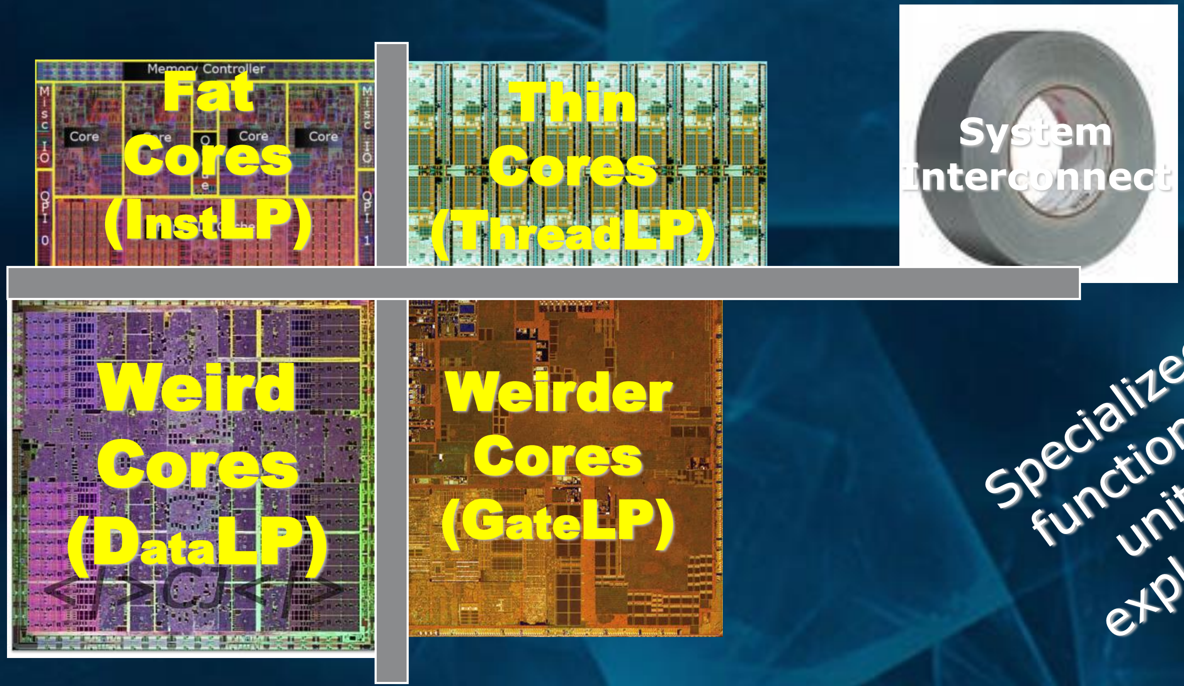
Tessellation Kernel Structure



Par Lab Architecture

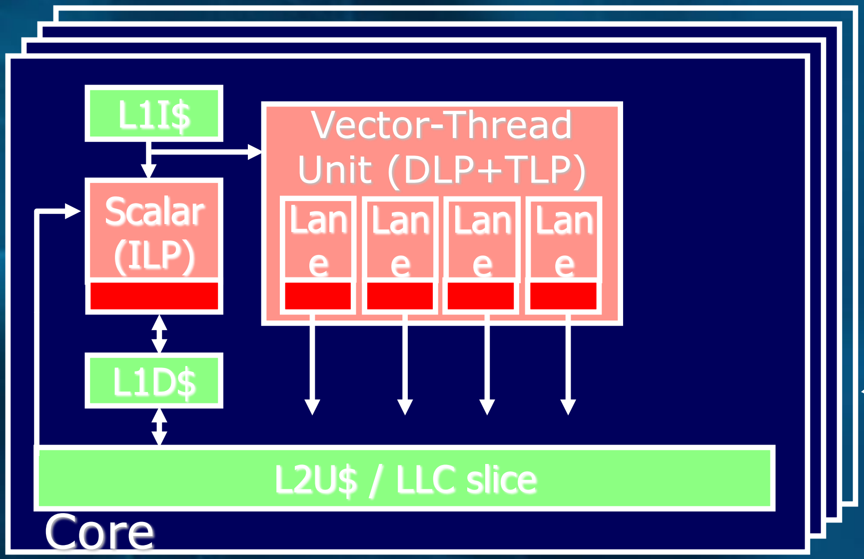
Microsoft Research

- Architect 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

Specialized functional units exploiting GLP



"Micro" TLP within core

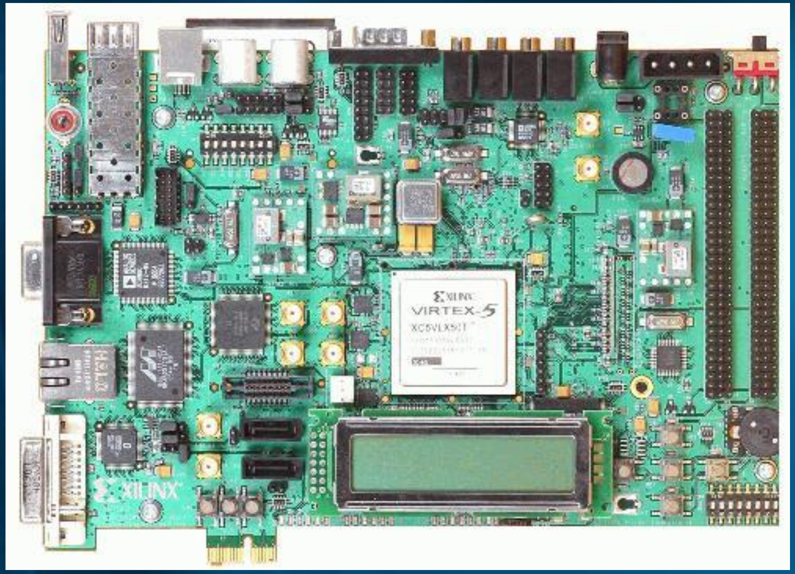
"Macro" TLP across cores

...instead, one type of multi-paradigm core

RAMP Gold

Microsoft Research

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



Software Simulator	$2,000	0.1 - 1	1
RAMP Gold	$2,000 + $750	50 - 100	100

Par Lab's original research "bets"

Microsoft Research

- Software platform: data center + mobile client
- Let compelling applications drive research agenda
- 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

Par Lab Research Overview

Easy to write correct programs that run efficiently on manycore