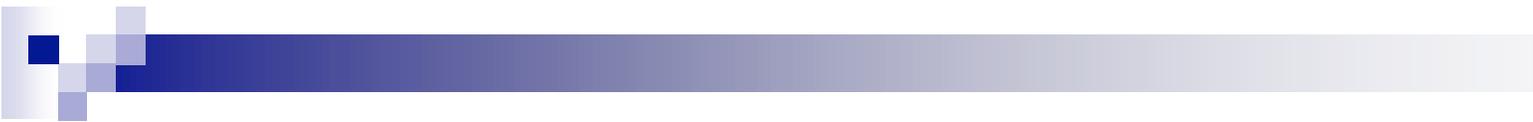


How to Hurt Scientific Productivity

David A. Patterson
Pardee Professor of Computer Science, U.C. Berkeley
President, Association for Computing Machinery

February, 2006



High Level Message

- Everything is changing; Old conventional wisdom is out
- We DESPERATELY need a new architectural solution for microprocessors based on parallelism
 - 21st Century target: systems that enhance scientific productivity
- Need to create a “watering hole” to bring everyone together to quickly find that solution
 - architects, language designers, application experts, numerical analysts, algorithm designers, programmers, ...



Computer Architecture Hurt #1: Aim High (and Ignore Amdahl's Law)

- Peak Performance Sells

- + Increases employment of computer scientists at companies trying to get larger fraction of peak

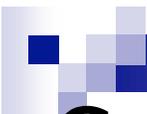
- Examples

- Very deep pipeline / very high clock rate
 - Relaxed write consistency
 - Out-Of-Order message delivery



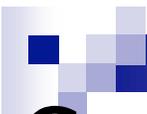
Computer Architecture Hurt #2: Promote Mystery (and Hide Thy Real Performance)

- Predictability suggests no sophistication
 - + If its unsophisticated, how can it be expensive?
- Examples
 - Out-of-order execution processors
 - Memory/disk controllers with secret prefetch algorithms
 - N levels of on-chip caches,
where $N \approx \lfloor (\text{Year} - 1975) / 10 \rfloor$



Computer Architecture Hurt #3: Be “Interesting” (and Have a Quirky Personality)

- Programmers enjoy a challenge
 - + Job security since must rewrite application with each new generation
- Examples
 - Message-passing clusters composed of shared address multiprocessors
 - Pattern sensitive interconnection networks
 - Computing using Graphical Processor Units
 - TLBs exceptions if access all cache memory on chip



Computer Architecture Hurt #4: Accuracy & Reliability are for Wimps (Speed Kills Competition)

- Don't waste resources on accuracy, reliability
 - + Probably blame Microsoft anyways
- Examples
 - Cray et al 754 Floating Point Format, yet not compliant, so get different results from desktop
 - No ECC on Memory of Virginia Tech Apple G5 cluster
 - "Error Free" intercommunication networks make error checking in messages "unnecessary"
 - No ECC on L2 Cache of Sun UltraSPARC 2

Alternatives to Hurting Productivity

- Aim High (& Ignore Amdahl's Law)?
 - No! Delivered productivity >> Peak performance
- Promote Mystery (& Hide Thy Real Performance)?
 - No! Promote a simple, understandable model of execution and performance
- Be "Interesting" (& Have a Quirky Personality)
 - No programming surprises!
- Accuracy & Reliability are for Wimps? (Speed Kills)
 - No! You're not going fast if you're headed in the wrong direction
- Computer designers neglected productivity in past
 - No excuse for 21st century computing to be based on untrustworthy, mysterious, I/O-starved, quirky HW where peak performance is king



Outline

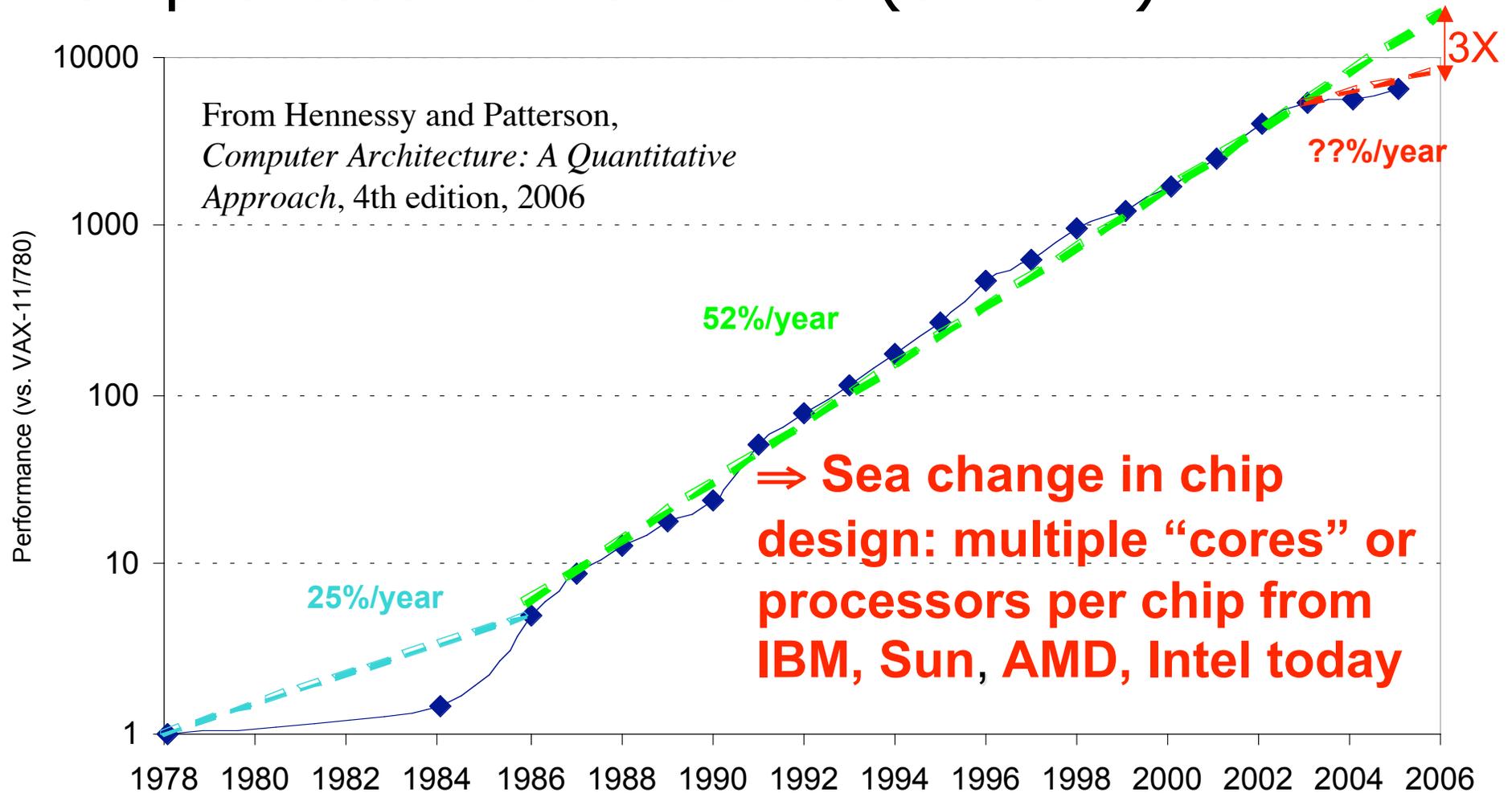
- Part I: How to Hurt Scientific Productivity
 - via Computer Architecture
- **Part II: A New Agenda for Computer Architecture**
 - 1st Review Conventional Wisdom (New & Old) in Technology and Computer Architecture
 - 21st century kernels, New classifications of apps and architecture
- **Part III: A “Watering Hole” for Parallel Systems Exploration**
 - **Research Accelerator for Multiple Processors**



Conventional Wisdom (CW) in Computer Architecture

- Old CW: Power is free, Transistors expensive
- New CW: “Power wall” Power expensive, Xtors free
(Can put more on chip than can afford to turn on)
- Old: Multiplies are slow, Memory access is fast
- New: “Memory wall” Memory slow, multiplies fast
(200 clocks to DRAM memory, 4 clocks for FP multiply)
- Old : Increasing Instruction Level Parallelism via compilers,
innovation (Out-of-order, speculation, VLIW, ...)
- New CW: “ILP wall” diminishing returns on more ILP
- New: Power Wall + Memory Wall + ILP Wall = Brick Wall
 - Old CW: Uniprocessor performance 2X / 1.5 yrs
 - New CW: Uniprocessor performance only 2X / 5 yrs?

Uniprocessor Performance (SPECint)

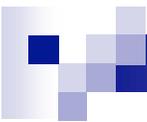


- **VAX** : 25%/year 1978 to 1986
- **RISC + x86**: 52%/year 1986 to 2002
- **RISC + x86**: ??%/year 2002 to present



21st Century Computer Architecture

- Old CW: Since cannot know future programs, find set of old programs to evaluate designs of computers for the future
 - E.g., SPEC2006
- What about parallel codes?
 - Few available, tied to old models, languages, architectures, ...
- New approach: Design computers of future for numerical methods important in future
- Claim: key methods for next decade are 7 dwarves (+ a few), so design for them!
 - Representative codes may vary over time, but these numerical methods will be important for > 10 years



Phillip Colella's "Seven dwarfs"
High-end simulation in the physical sciences = 7 numerical methods:

1. Structured Grids (including locally structured grids, e.g. Adaptive Mesh Refinement)
 2. Unstructured Grids
 3. Fast Fourier Transform
 4. Dense Linear Algebra
 5. Sparse Linear Algebra
 6. Particles
 7. Monte Carlo
- If add 4 for embedded, covers all 41 EEMBC benchmarks
 8. Search/Sort
 9. Filter
 10. Combinational logic
 11. Finite State Machine
 - Note: Data sizes (8 bit to 32 bit) and types (integer, character) differ, but algorithms the same

Well-defined targets from algorithmic, software, and architecture standpoint

Slide from "Defining Software Requirements for Scientific Computing", Phillip Colella, 2004



6/11 Dwarves Covers 24/30 SPEC2006

■ SPECfp

- 8 Structured grid
 - 3 using Adaptive Mesh Refinement
- 2 Sparse linear algebra
- 2 Particle methods
- 5 TBD: Ray tracer, Speech Recognition, Quantum Chemistry, Lattice Quantum Chromodynamics (many kernels inside each benchmark?)

■ SPECint

- 8 Finite State Machine
- 2 Sorting/Searching
- 2 Dense linear algebra (data type differs from dwarf)
- 1 TBD: 1 C compiler (many kernels?)



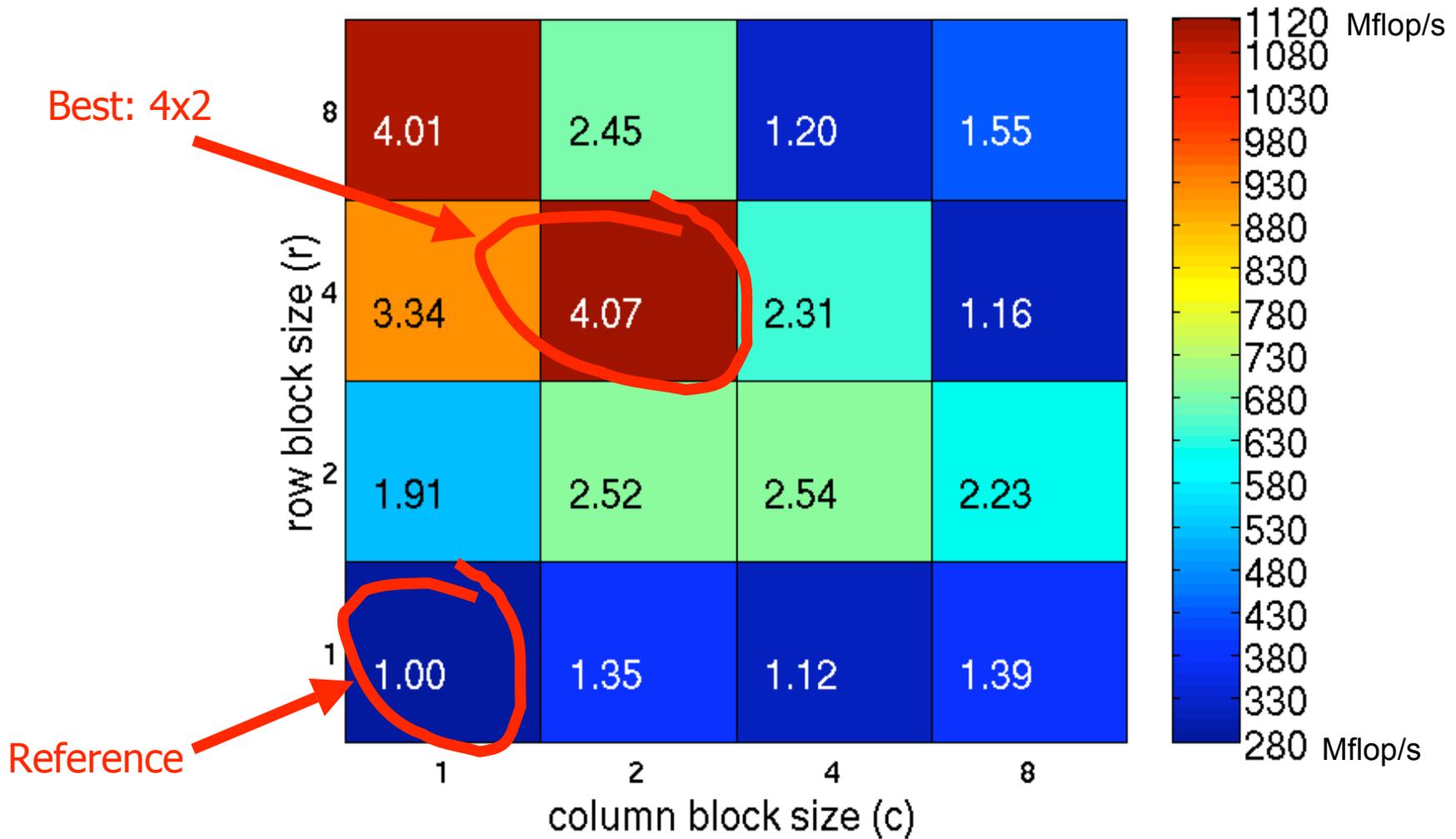
21st Century Code Generation

- Old CW: Takes a decade for compilers to introduce an architecture innovation
- New approach: “**Auto-tuners**” 1st run variations of program on computer to find best combinations of optimizations (blocking, padding, ...) and algorithms, then produce C code to be compiled for *that* computer
 - E.g., PHiPAC (Portable High Performance Ansi C), Atlas (BLAS), Sparsity (Sparse linear algebra), Spiral (DSP), FFT-W
 - Can achieve large speedup over conventional compiler
- One Auto-tuner per dwarf?
 - Exist for Dense Linear Algebra, Sparse Linear Algebra, Spectral

Sparse Matrix – Search for Blocking

for finite element problem [Im, Yelick, Vuduc, 2005]

900 MHz Itanium 2, Intel C v8: ref=275 Mflop/s





21st Century Classification

- Old CW:
 - SISD vs. SIMD vs. MIMD
- 3 “new” measures of parallelism
 - Size of Operands
 - Style of Parallelism
 - Amount of Parallelism



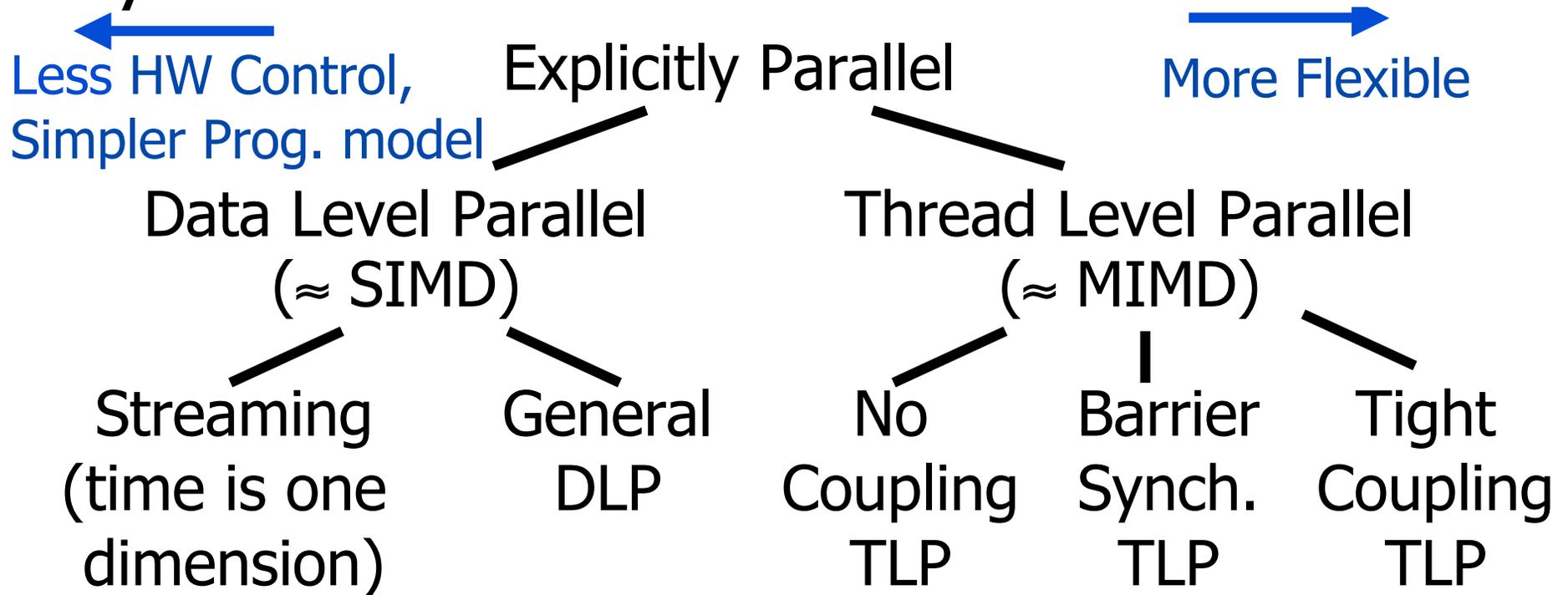
Operand Size and Type

Programmer should be able to specify data size, type independent of algorithm

- 1 bit (**Boolean***)
- 8 bits (Integer, ASCII)
- 16 bits (Integer, DSP fixed pt, **Unicode***)
- 32 bits (Integer, SP Fl. Pt., **Unicode***)
- 64 bits (Integer, DP Fl. Pt.)
- 128 bits (**Integer***, **Quad Precision Fl. Pt.***)
- 1024 bits (**Crypto***)

* Not supported well in most programming languages and optimizing compilers

Style of Parallelism

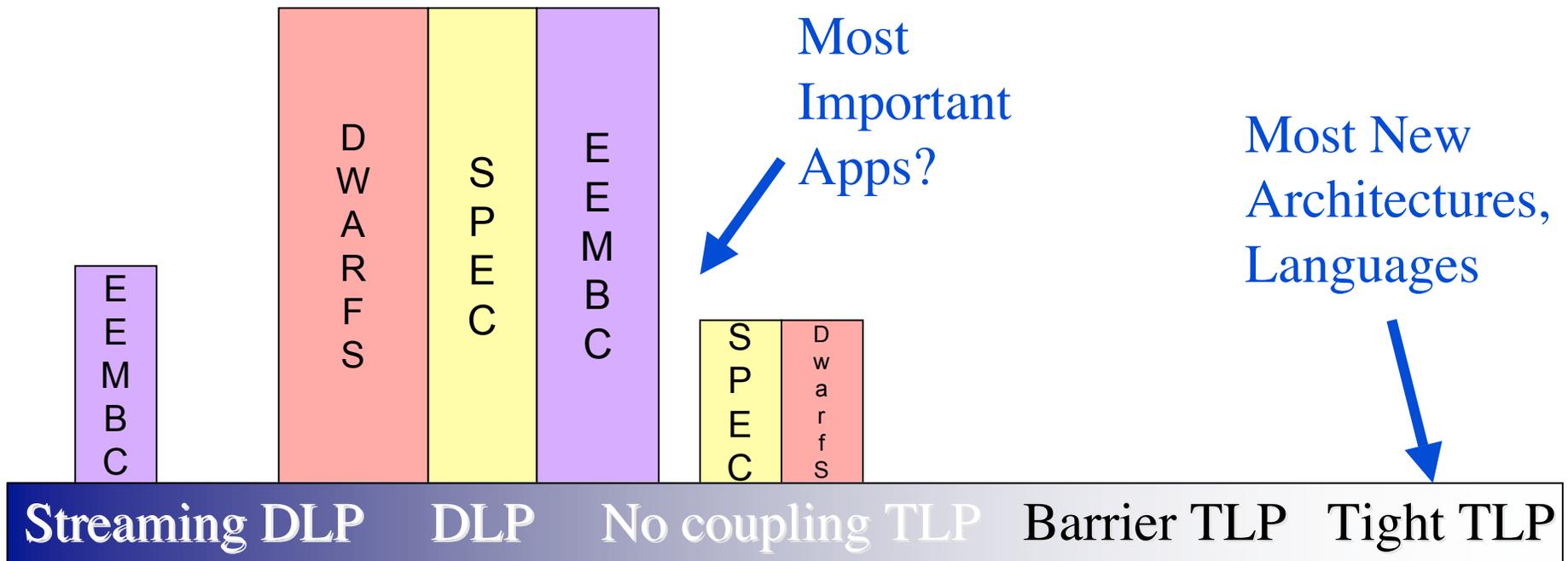


Programmer wants code to run on as many parallel architectures as possible so (if possible)

Architect wants to run as many different types of parallel programs as possible so

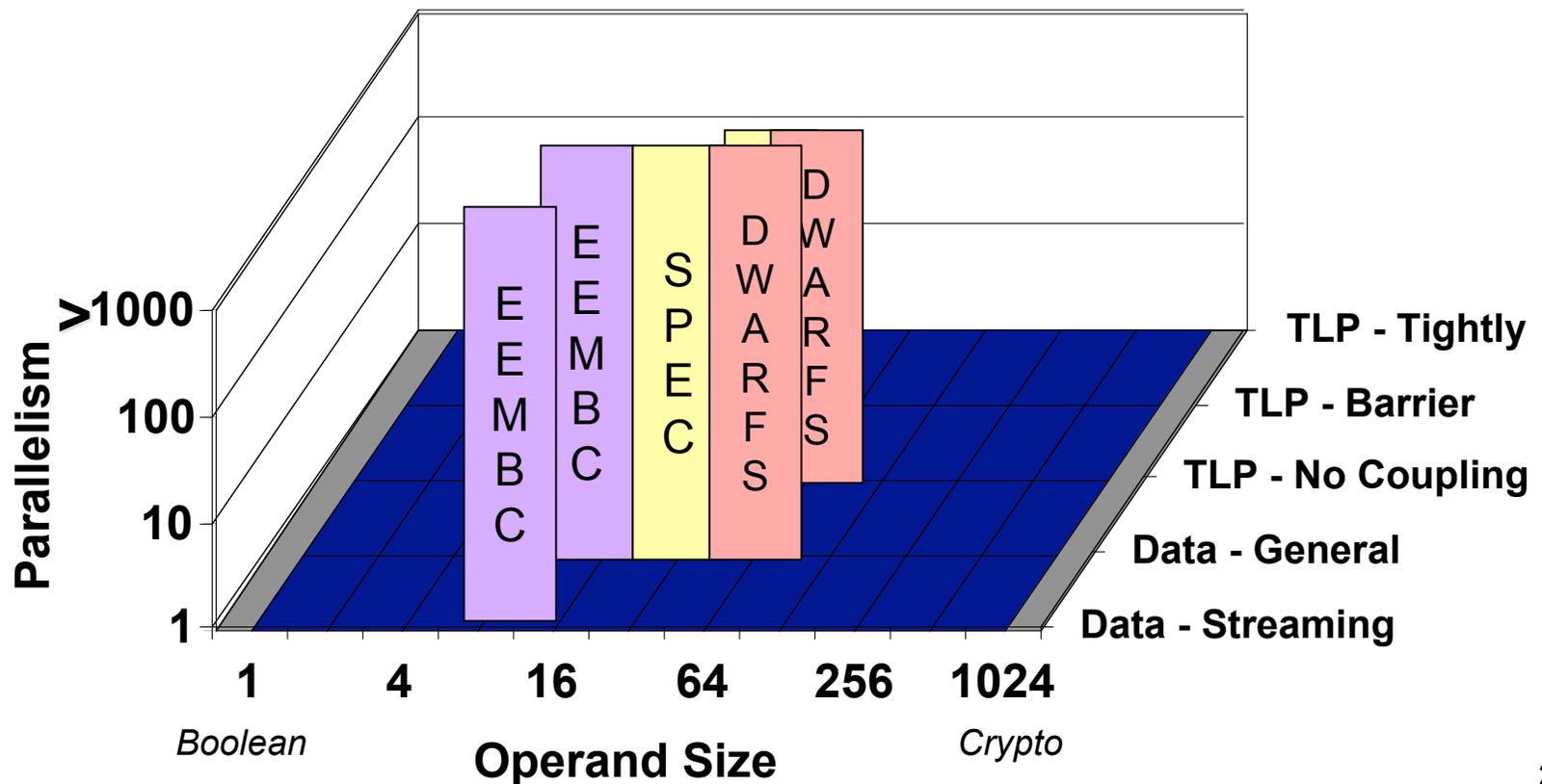
Parallel Framework – Apps (so far)

- Original 7 dwarves: 6 data parallel, 1 no coupling TLP
- Bonus 4 dwarves: 2 data parallel, 2 no coupling TLP
- EEMBC (Embedded): Stream 10, DLP 19, Barrier TLP 2
- SPEC (Desktop): 14 DLP, 2 no coupling TLP



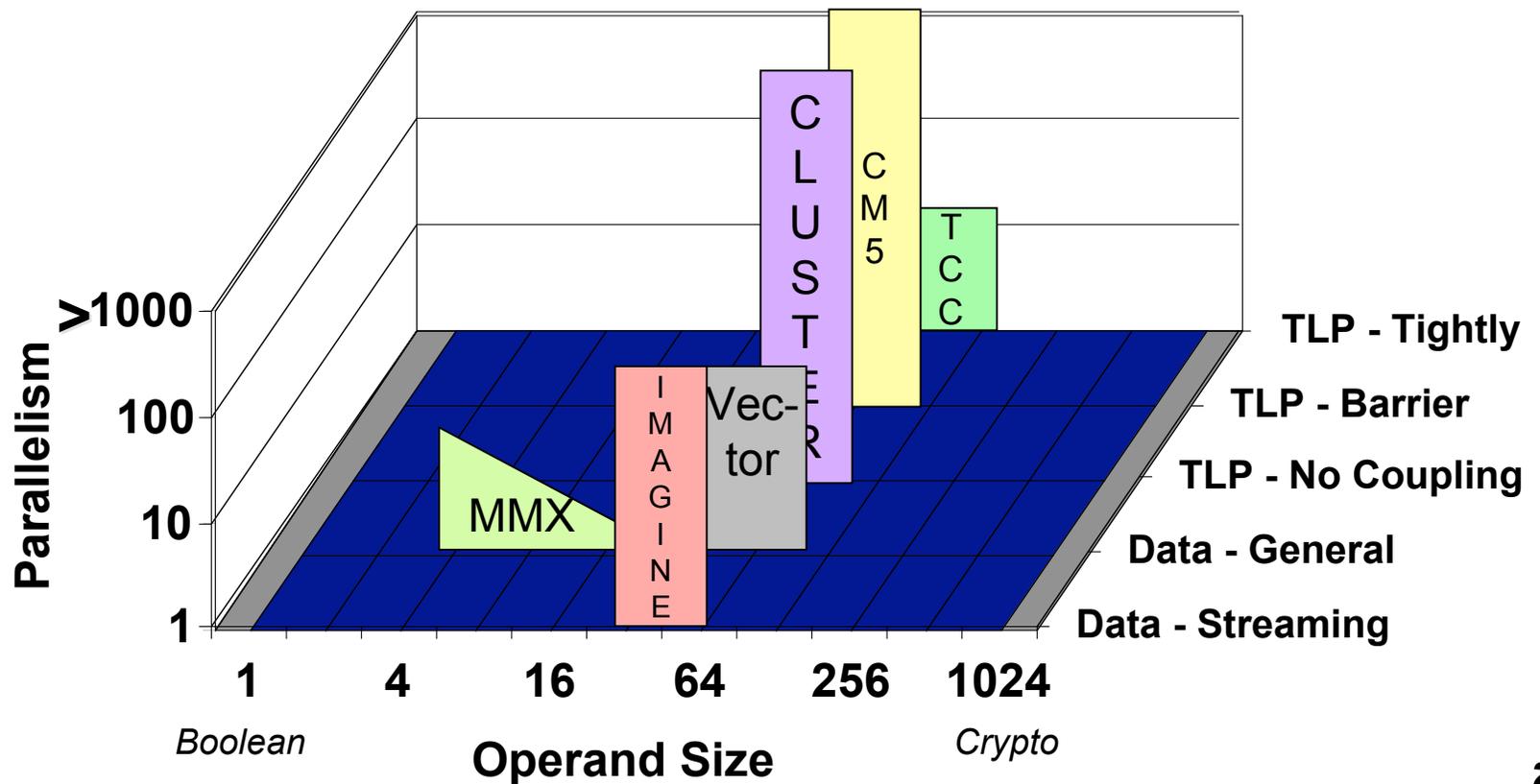
New Parallel Framework

- Given natural operand size and level of parallelism, how parallel is computer or how must parallelism available in application?
- Proposed Parallel Framework for Arch and Apps



Parallel Framework - Architecture

- Examples of good architectural matches to each style





Outline

- **Part I: How to Hurt Scientific Productivity**
 - via Computer Architecture
- **Part II: A New Agenda for Computer Architecture**
 - 1st Review Conventional Wisdom (New & Old) in Technology and Computer Architecture
 - 21st century kernels, New classifications of apps and architecture
- **Part III: A “Watering Hole” for Parallel Systems Exploration**
 - **Research Accelerator for Multiple Processors**
- **Conclusion**



Problems with Sea Change

1. Algorithms, Programming Languages, Compilers, Operating Systems, Architectures, Libraries, ... not ready for 1000 CPUs / chip
2. Only companies can build HW, and it takes years
 - \$M mask costs, \$M for ECAD tools, GHz clock rates, >100M transistors
3. Software people don't start working hard until hardware arrives
 - 3 months after HW arrives, SW people list everything that must be fixed, then we all wait 4 years for next iteration of HW/SW
4. How get 1000 CPU systems in hands of researchers to innovate in timely fashion on in algorithms, compilers, languages, OS, architectures, ... ?
5. Avoid waiting years between HW/SW iterations?

Build Academic MPP from FPGAs

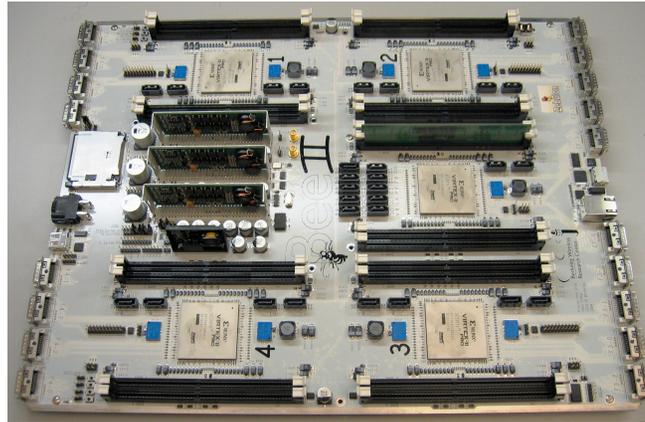
- As ≈ 25 CPUs will fit in Field Programmable Gate Array (FPGA), 1000-CPU system from ≈ 40 FPGAs?
 - 16 32-bit simple “soft core” RISC at 150MHz in 2004 (Virtex-II)
 - FPGA generations every 1.5 yrs; $\approx 2X$ CPUs, $\approx 1.2X$ clock rate
- HW research community does logic design (“gate shareware”) to create out-of-the-box, MPP
 - E.g., 1000 processor, standard ISA binary-compatible, 64-bit, cache-coherent supercomputer @ ≈ 100 MHz/CPU in 2007
 - RAMPants: Arvind (MIT), [Krste Asanović](#) (MIT), Derek Chiou (Texas), James Hoe (CMU), [Christos Kozyrakis](#) (Stanford), Shih-Lien Lu (Intel), Mark Oskin (Washington), [David Patterson](#) (Berkeley, Co-PI), Jan Rabaey (Berkeley), and John Wawrzynek (Berkeley, PI)
- “Research Accelerator for Multiple Processors”

RAMP 1 Hardware

- Completed Dec. 2004 (14x17 inch 22-layer PCB)

Board:

5 Virtex II FPGAs, 18 banks DDR2-400 memory, 20 10GigE conn.



1.5W / computer,
5 cu. in. /computer,
\$100 / computer

Box:

8 compute modules in
8U rack mount chassis

1000 CPUs :
≈ 1.5 KW,
≈ _ rack,
≈ \$100,000



BEE2: Berkeley Emulation Engine 2

By John Wawrzynek and Bob Brodersen with
students Chen Chang and Pierre Droz

RAMP Milestones

Name	Goal	Target	CPUs	Details
Red (Stanford)	Get Started	1H06	8 PowerPC 32b hard cores	Transactional memory SMP
Blue (Cal)	Scale	2H06	≈1000 32b soft (Microblaze)	Cluster, MPI
White (All)	Full Features	1H07?	128? soft 64b, Multiple commercial ISAs	CC-NUMA, shared address, deterministic, debug/monitor
2.0	3 rd party sells it	2H07?	4X CPUs of '04 FPGA	New '06 FPGA, new board

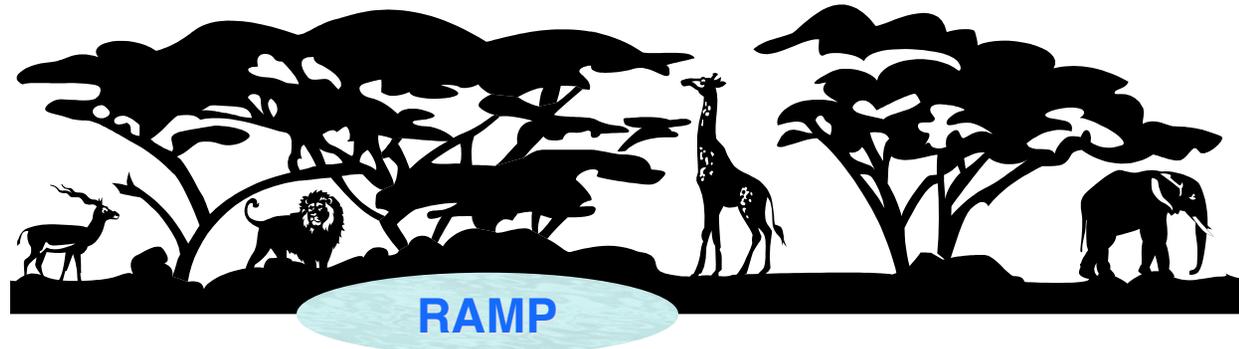
Can RAMP keep up?

- FGPA generations: 2X CPUs / 18 months
 - 2X CPUs / 24 months for desktop microprocessors
- 1.1X to 1.3X performance / 18 months
 - 1.2X? / year per CPU on desktop?
- However, goal for RAMP is **accurate system emulation**, not to be the real system
 - Goal is accurate target performance, parameterized reconfiguration, extensive monitoring, reproducibility, cheap (**like a simulator**) while being credible and fast enough to emulate 1000s of OS and apps in parallel (**like hardware**)

RAMP + Auto-tuners = Promised land?

- Auto-tuners in reaction to fixed, hard to understand hardware
- RAMP enables perpendicular exploration
- For each algorithm, how can the architecture be modified to achieve maximum performance given the resource limitations (e.g., bandwidth, cache-sizes, ...)
- Auto-tuning searches can focus on comparing different algorithms for each dwarf rather than also spending time massaging computer quirks

Multiprocessing Watering Hole



Parallel file system Dataflow language/computer Data center in a box
Fault insertion to check dependability Router design Compile to FPGA
Flight Data Recorder Security enhancements Transactional Memory
Internet in a box 128-bit Floating Point Libraries Parallel languages

- Killer app: \approx All CS Research, Advanced Development
- RAMP attracts many communities to shared artifact
⇒ Cross-disciplinary interactions
⇒ Ramp up innovation in multiprocessing
- RAMP as next Standard Research/AD Platform?
(e.g., VAX/BSD Unix in 1980s, Linux/x86 in 1990s)

Conclusion: [1 / 2]

Alternatives to Hurting Productivity

- Delivered productivity >> Peak performance
- Promote a simple, understandable model of execution and performance
- No programming surprises!
- You're not going fast if you're going the wrong way

Use Programs of Future to design Computers, Languages, ... of the Future

- 7 + 5? Dwarves, Auto-Tuners, RAMP
- Although architect's, language designers focusing toward right, most dwarves are toward left

Conclusions [2 / 2]

- **Research Accelerator for Multiple Processors**
- **Carpe Diem: Researchers need it ASAP**
 - FPGAs ready, and getting better
 - Stand on shoulders vs. toes: standardize on Berkeley FPGA platforms (BEE, BEE2) by Wawrzynek et al
 - Architects aid colleagues via gateware
- **RAMP accelerates HW/SW generations**
 - System emulation + good accounting vs. FPGA computer
 - Emulate, Trace, Reproduce anything; Tape out every day
- **“Multiprocessor Research Watering Hole”**

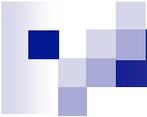
ramp up research in multiprocessing via common research platform ⇒ innovate across fields ⇒ hasten sea change from sequential to parallel computing

Acknowledgments

- Material comes from discussions on new directions for architecture with:
 - Professors [Krste Asanović](#) (MIT), Raz Bodik, Jim Demmel, Kurt Kuetzer, John Wawrzynek, and [Kathy Yelick](#)
 - LBNL discussants Parry Husbands, Bill Kramer, Lenny Oliner, and [John Shalf](#)
 - UCB Grad students Joe Gebis and Sam Williams
- RAMP based on work of RAMP Developers:
 - Arvind (MIT), [Krste Asanović](#) (MIT), Derek Chiou (Texas), James Hoe (CMU), [Christos Kozyrakis](#) (Stanford), Shih-Lien Lu (Intel), Mark Oskin (Washington), [David Patterson](#) (Berkeley, Co-PI), Jan Rabaey (Berkeley), and John Wawrzynek (Berkeley, PI)
- See ramp.eecs.berkeley.edu



Backup Slides



Summary of Dwarves (so far)

- Original 7: 6 data parallel, 1 no coupling TLP
- Bonus 4: 2 data parallel, 2 no coupling TLP
 - To Be Done: FSM
- EEMBC (Embedded): Stream 10, DLP 19
 - Barrier (2), 11 more to characterize
- SPEC (Desktop): 14 DLP, 2 no coupling TLP
 - 6 dwarves cover 24/30; To Be Done: 8 FSM, 6 Big SPEC
- Although architect's focusing toward right, most dwarves are toward left

Streaming DLP	DLP	No coupling TLP	Barrier TLP	Tight TLP
---------------	-----	-----------------	-------------	-----------

Supporters *(wrote letters to NSF)*

- Gordon Bell (Microsoft)
- Ivo Bolsens (Xilinx CTO)
- Norm Jouppi (HP Labs)
- Bill Kramer (NERSC/LBL)
- Craig Mundie (MS CTO)
- G. Papadopoulos (Sun CTO)
- Justin Rattner (Intel CTO)
- Ivan Sutherland (Sun Fellow)
- Chuck Thacker (Microsoft)
- Kees Vissers (Xilinx)
- Doug Burger (Texas)
- Bill Dally (Stanford)
- Carl Ebeling (Washington)
- Susan Eggers (Washington)
- Steve Keckler (Texas)
- Greg Morrisett (Harvard)
- Scott Shenker (Berkeley)
- Ion Stoica (Berkeley)
- **Kathy Yelick (Berkeley)**

RAMP Participants: Arvind (MIT), **Krste Asanović (MIT)**, Derek Chiou (Texas), James Hoe (CMU), **Christos Kozyrakis (Stanford)**, Shih-Lien Lu (Intel), Mark Oskin (Washington), **David Patterson (Berkeley, Co-PI)**, Jan Rabaey (Berkeley), and John Wawrzynek (Berkeley, PI)

RAMP FAQ

- Q: What about power, cost, space in RAMP?
- A:
 - 1.5 watts per computer
 - \$100-\$200 per computer
 - 5 cubic inches per computer
 - 1000 computers for \$100k to \$200k, 1.5 KW, 1/3 rack
- Using very slow clock rate, very simple CPUs, and very large FPGAs

A decorative graphic in the top left corner, featuring a grid of squares in shades of blue and grey, followed by a horizontal bar with a gradient from dark blue to light grey.

RAMP FAQ

- Q: How will FPGA clock rate improve?
- A1: 1.1X to 1.3X / 18 months
 - Note that clock rate now going up slowly on desktop
- A2: Goal for RAMP is system emulation, not to be the real system
 - Hence, value accurate accounting of target clock cycles, parameterized design (Memory BW, network BW, ...), monitor, debug over performance
 - Goal is just fast enough to emulate OS, app in parallel

A decorative graphic in the top left corner consists of a grid of squares in shades of blue and grey, followed by a horizontal bar with a blue-to-white gradient.

RAMP FAQ

- Q: What about power, cost, space in RAMP?
- A:
 - 1.5 watts per computer
 - \$100-\$200 per computer
 - 5 cubic inches per computer
- Using very slow clock rate, very simple CPUs in a very large FPGA (RAMP blue)

RAMP FAQ

- Q: How can many researchers get RAMPs?
- A1: RAMP 2.0 to be available for purchase at low margin from 3rd party vendor
- A2: Single board RAMP 2.0 still interesting as FPGA 2X CPUs/18 months
 - RAMP 2.0 FPGA two generations later than RAMP 1.0, so 256? simple CPUs per board vs. 64?

Parallel FAQ

- Q: Won't the circuit or processing guys solve CPU performance problem for us?
- A1: No. More transistors, but can't help with ILP wall, and power wall is close to fundamental problem
 - Memory wall could be lowered some, but hasn't happened yet commercially
- A2: One time jump. IBM using "strained silicon" on Silicon On Insulator to increase electron mobility (Intel doesn't have SOI)
 - ⇒ clock rate↑ or leakage power↓
 - Continue making rapid semiconductor investment?

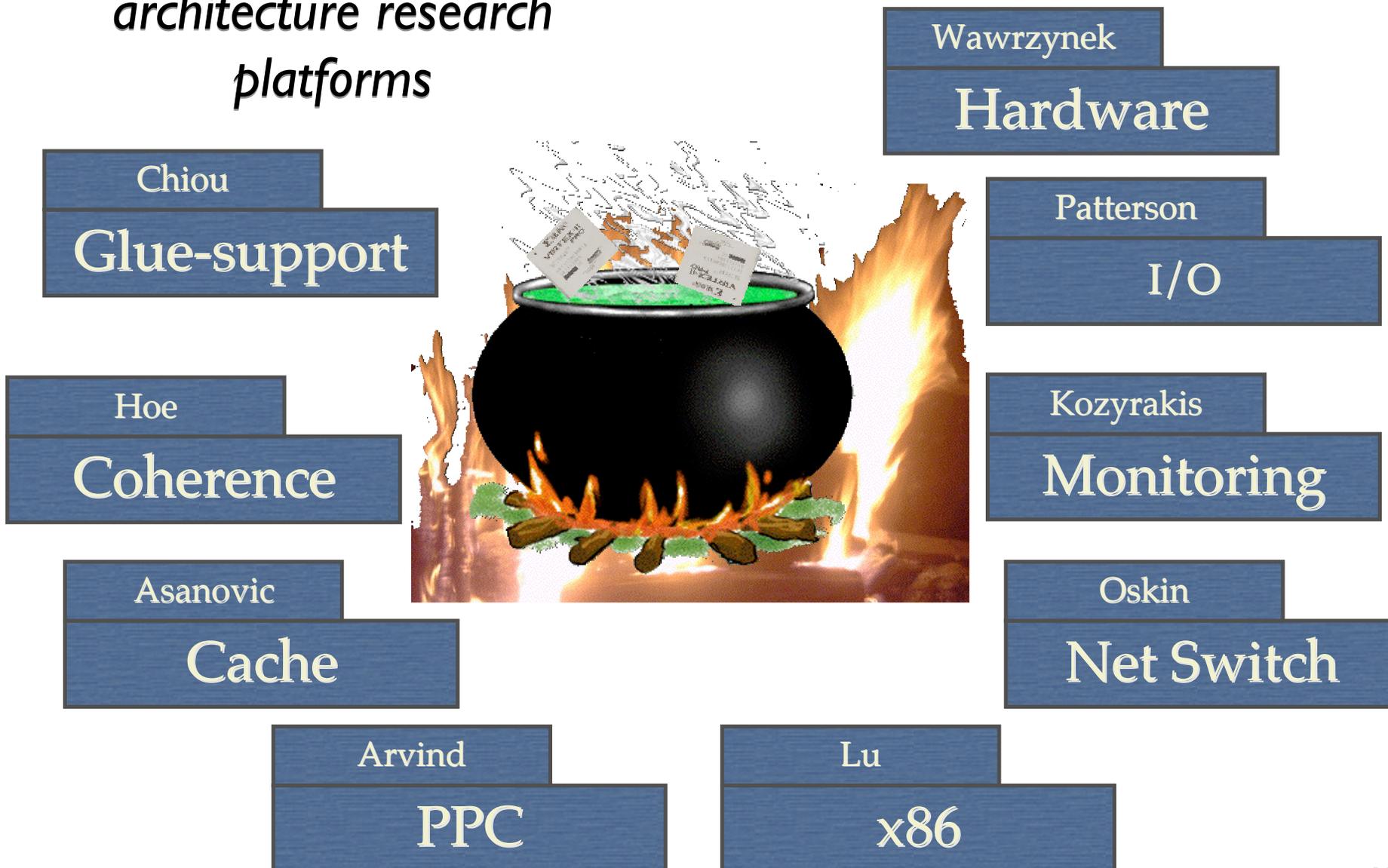
Parallel FAQ

- Q: How afford 2 processors if power is the problem?
- A: Simpler core, lower voltage and frequency
 - Power \approx Capacitance \times Volt² \times Frequency : $0.85^4 \approx 0.5$
 - Also, single complex CPU inefficient in transistors, power

RAMP Development Plan

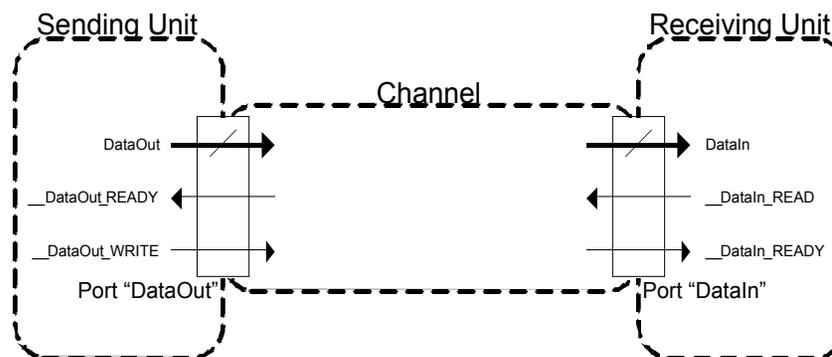
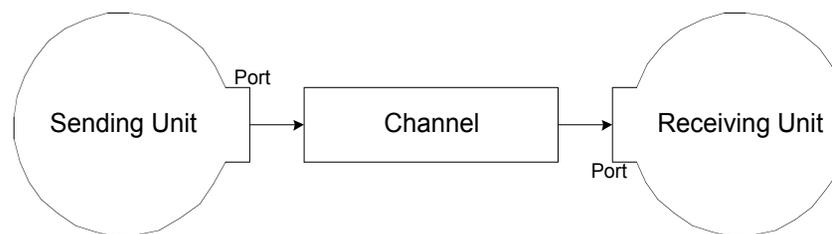
1. Distribute systems internally for RAMP 1 development
 - Xilinx agreed to pay for production of a set of modules for initial contributing developers and first full RAMP system
 - Others could be available if can recover costs
2. Release publicly available out-of-the-box MPP emulator
 - Based on standard ISA (IBM Power, Sun SPARC, ...) for binary compatibility
 - Complete OS/libraries
 - **Locally modify RAMP as desired**
3. Design next generation platform for RAMP 2
 - Base on 65nm FPGAs (2 generations later than Virtex-II)
 - Pending results from RAMP 1, Xilinx will cover hardware costs for initial set of RAMP 2 machines
 - **Find 3rd party to build and distribute systems (at *near-cost*), open source RAMP gateware and software**
 - **Hope RAMP 3, 4, ... self-sustaining**
- NSF/CRI proposal pending to help support effort
 - 2 full-time staff (one HW/gateware, one OS/software)
 - Look for grad student support at 6 RAMP universities from industrial donations

*the stone soup of
architecture research
platforms*



Gateway Design Framework

- Design composed of units that send messages over channels via ports
- Units (10,000 + gates)
 - CPU + L1 cache, DRAM controller....
- Channels (\approx FIFO)
 - Lossless, point-to-point, unidirectional, in-order message delivery...



Gateway Design Framework

- Insight: almost every large building block fits inside FPGA today
 - what doesn't is between chips in real design
- Supports both cycle-accurate emulation of detailed parameterized machine models and rapid functional-only emulations
- Carefully counts for *Target Clock Cycles*
- Units in any hardware design language (will work with Verilog, VHDL, BlueSpec, C, ...)
- RAMP Design Language (RDL) to describe plumbing to connect units in

Quick Sanity Check

- BEE2 uses old FPGAs (Virtex II), 4 banks DDR2-400/cpu
- 16 32-bit Microblazes per Virtex II FPGA, 0.75 MB memory for caches
 - 32 KB direct mapped Icache, 16 KB direct mapped Dcache
- Assume 150 MHz, CPI is 1.5 (4-stage pipe)
 - I\$ Miss rate is 0.5% for SPECint2000
 - D\$ Miss rate is 2.8% for SPECint2000, 40% Loads/stores
- $BW\ need/CPU = 150/1.5 * 4B * (0.5\% + 40\% * 2.8\%)$
 $= 6.4\ MB/sec$
- $BW\ need/FPGA = 16 * 6.4 = 100\ MB/s$
- $Memory\ BW/FPGA = 4 * 200\ MHz * 2 * 8B = 12,800\ MB/s$
- Plenty of BW for tracing, ...

RAMP FAQ on ISAs

- Which ISA will you pick?
 - Goal is replaceable ISA/CPU L1 cache, rest infrastructure unchanged (L2 cache, router, memory controller, ...)
- What do you want from a CPU?
 - Standard ISA (binaries, libraries, ...), simple (area), 64-bit (coherency), DP Fl.Pt. (apps)
 - Multithreading? As an option, but want to get to 1000 independent CPUs
- When do you need it? 3Q06
- RAMP people port my ISA , fix my ISA?
 - Our plates are full already
 - Type A vs. Type B gateware
 - Router, Memory controller, Cache coherency, L2 cache, Disk module, protocol for each
 - Integration, testing

Handicapping ISAs

- Got it: Power 405 (32b), SPARC v8 (32b),
Xilinx Microblaze (32b)
- Very Likely: SPARC v9 (64b)
- Likely: IBM Power 64b
- Probably (haven't asked): MIPS32, MIPS64
- No: x86, x86-64
 - But Derek Chiou of UT looking at x86 binary translation
- We'll sue: ARM
 - But pretty simple ISA & MIT has good lawyers

Related Approaches (1)

- **Quickturn, Axis, IKOS, Thara:**
 - FPGA- or special-processor based gate-level hardware emulators
 - Synthesizable HDL is mapped to array for cycle and bit-accurate netlist emulation
 - RAMP's emphasis is on emulating high-level architecture behaviors
 - Hardware and supporting software provides architecture-level abstractions for modeling and analysis
 - Targets architecture and software research
 - Provides a spectrum of tradeoffs between speed and accuracy/precision of emulation
- **RPM at USC in early 1990's:**
 - Up to only 8 processors
 - Only the memory controller implemented with configurable logic

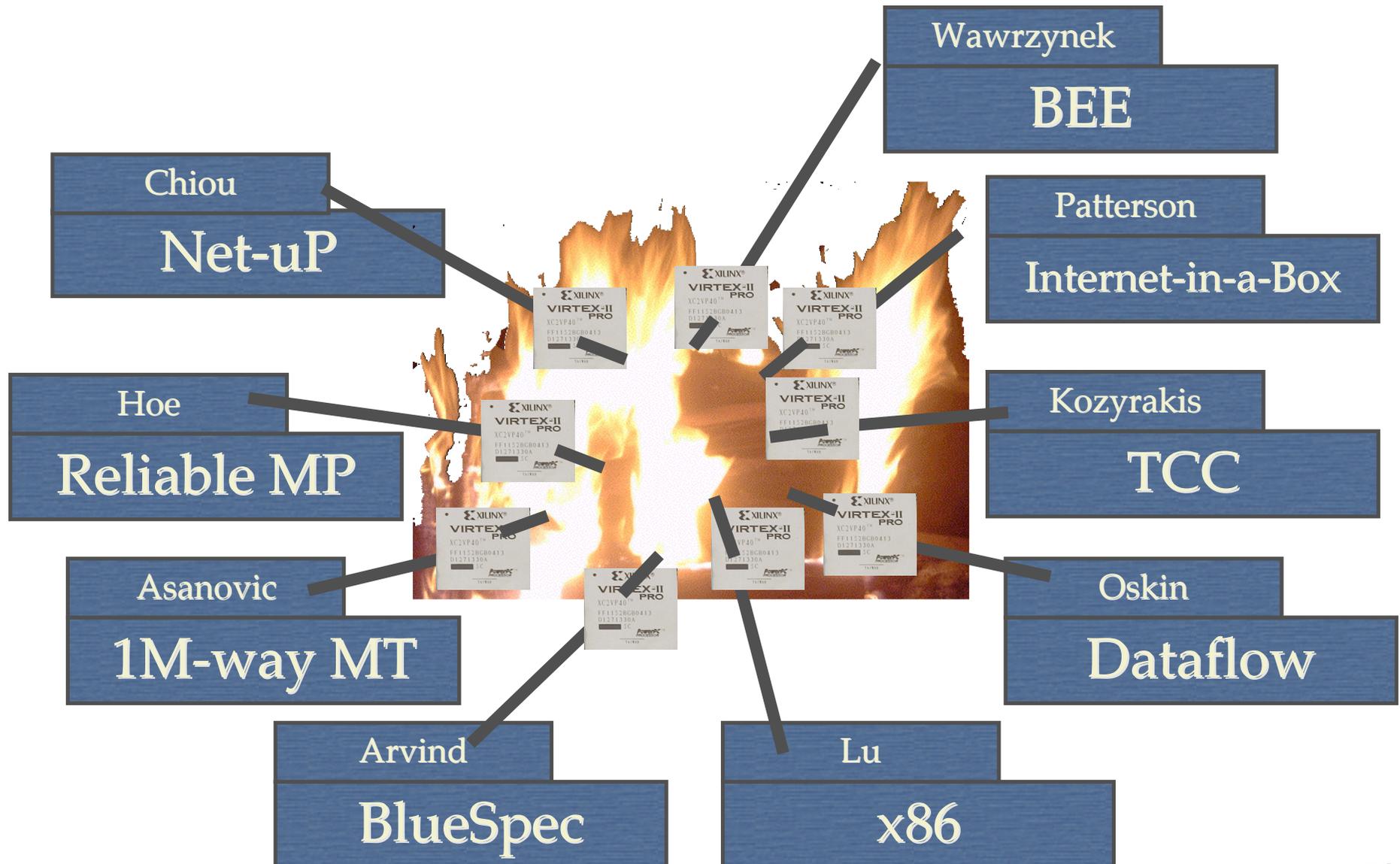
Related Approaches (2)

- Software Simulators
- Clusters (standard microprocessors)
- PlanetLab (distributed environment)
- Wisconsin Wind Tunnel (used CM-5 to simulate shared memory)

All suffer from some combination of:

- Slowness, inaccuracy, scalability, unbalanced computation/communication, target inflexibility

RAMP uses (internal)



RAMP Example: UT FAST

- 1MHz to 100MHz, cycle-accurate, full-system, multiprocessor simulator
 - Well, not quite that fast right now, but we are using embedded 300MHz PowerPC 405 to simplify
- X86, boots Linux, Windows, targeting 80486 to Pentium M-like designs
 - Heavily modified Bochs, supports instruction trace and rollback
- Working on “superscalar” model
 - Have straight pipeline 486 model with TLBs and caches
- Statistics gathered in hardware
 - Very little if any probe effect
- Work started on tools to semi-automate micro-architectural and ISA level exploration
 - Orthogonality of models makes both simpler

A decorative graphic in the top left corner consists of a grid of squares in shades of blue and grey, followed by a horizontal bar that transitions from dark blue to light grey.

Example: Transactional Memory

- Processors/memory hierarchy that support transactional memory
- Hardware/software infrastructure for performance monitoring and profiling
 - Will be general for any type of event
- Transactional coherence protocol

A decorative graphic in the top left corner consists of a grid of squares in shades of blue and grey, followed by a horizontal bar that transitions from dark blue to light grey.

Example: PROTOFLEX

- Hardware/Software Co-simulation/test methodology
- Based on FLEXUS C++ full-system multiprocessor simulator
 - Can swap out individual components to hardware
- Used to create and test a non-block MSI invalidation-based protocol engine in hardware

A decorative graphic in the top left corner consists of a grid of squares in shades of blue and grey, followed by a horizontal bar that transitions from dark blue to light grey.

Example: Wavescalar Infrastructure

- Dynamic Routing Switch
- Directory-based coherency scheme and engine

Example RAMP App: "Internet in a Box"

- Building blocks also \Rightarrow Distributed Computing
- RAMP vs. Clusters (Emulab, PlanetLab)
 - Scale: RAMP $O(1000)$ vs. Clusters $O(100)$
 - Private use: \$100k \Rightarrow Every group has one
 - Develop/Debug: Reproducibility, Observability
 - Flexibility: Modify modules (SMP, OS)
 - Heterogeneity: Connect to diverse, real routers
- Explore via repeatable experiments as vary parameters, configurations vs. observations on single (aging) cluster that is often idiosyncratic

Conventional Wisdom (CW) in Scientific Programming

- Old CW: Programming is hard
- New CW: Parallel programming is really hard
 - 2 kinds of Scientific Programmers
 - Those using single processor
 - Those who can use up to 100 processors
 - Big steps for programmers
 - From 1 processor to 2 processors
 - From 100 processors to 1000 processors
 - Can computer architecture make many processors look like fewer processors, ideally one?
- Old CW: Who cares about I/O in Supercomputing?
- New CW: Supercomputing
= Massive data + Massive Computation

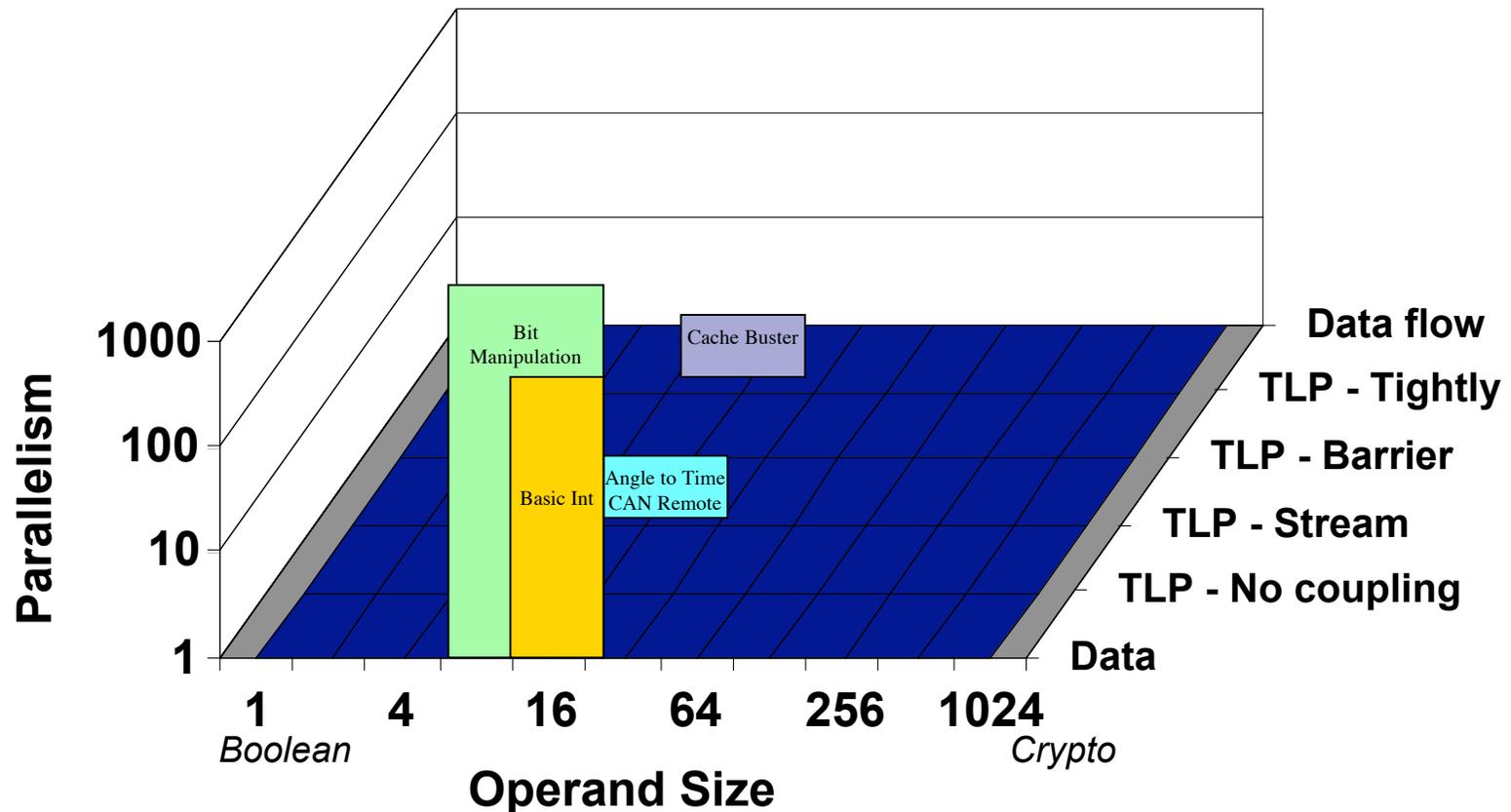
A decorative header at the top left consists of a grid of squares in shades of blue and grey, followed by a horizontal bar that transitions from dark blue to light grey.

Size of Parallel Computer

- What parallelism achievable with good or bad architectures, good or bad algorithms?
 - 32-way: anything goes
 - 100-way: good architecture and bad algorithms
or bad architecture and good algorithms
 - 1000-way: good architecture and good algorithm

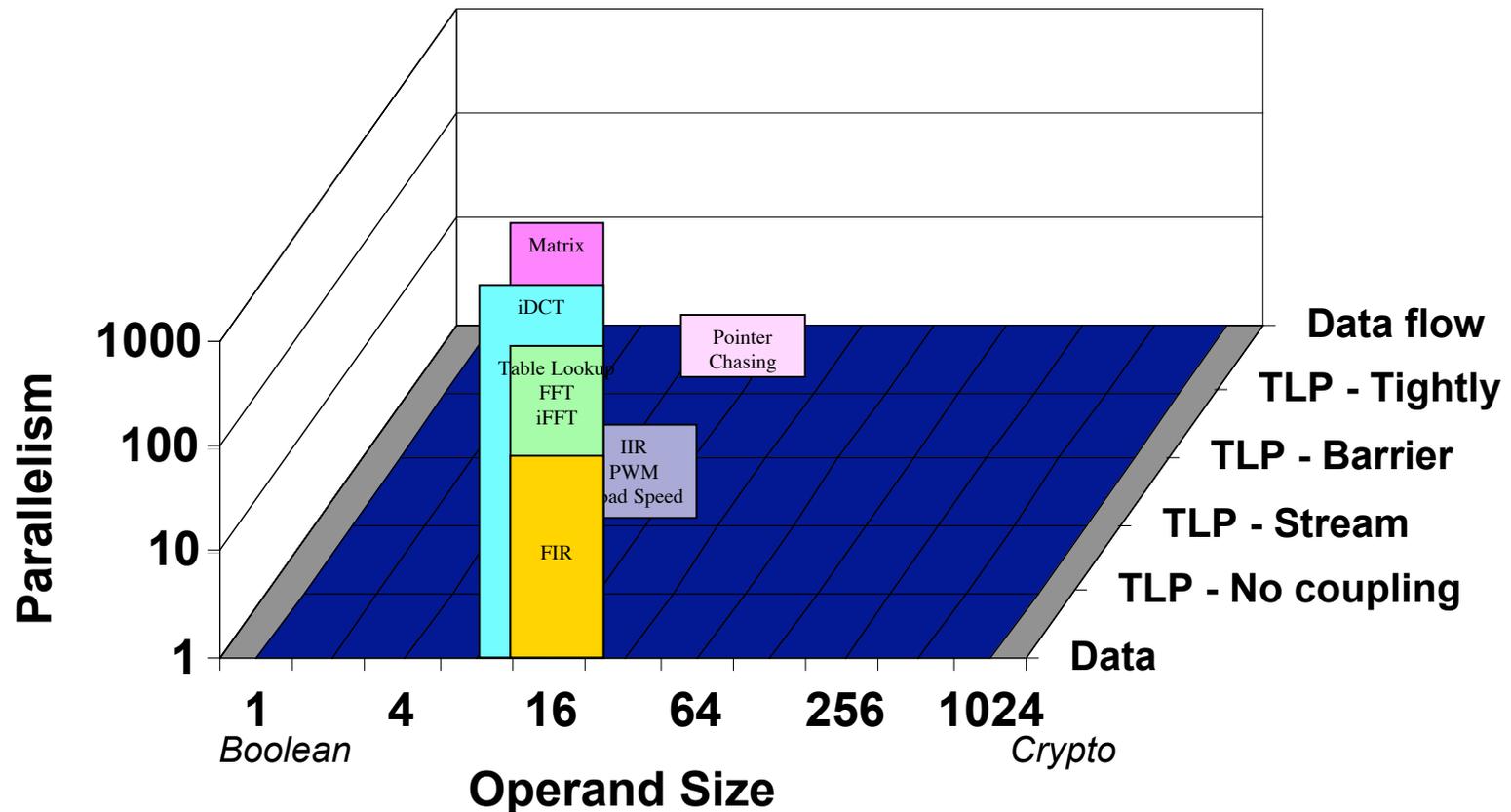
Parallel Framework - Benchmarks

- EEMBC



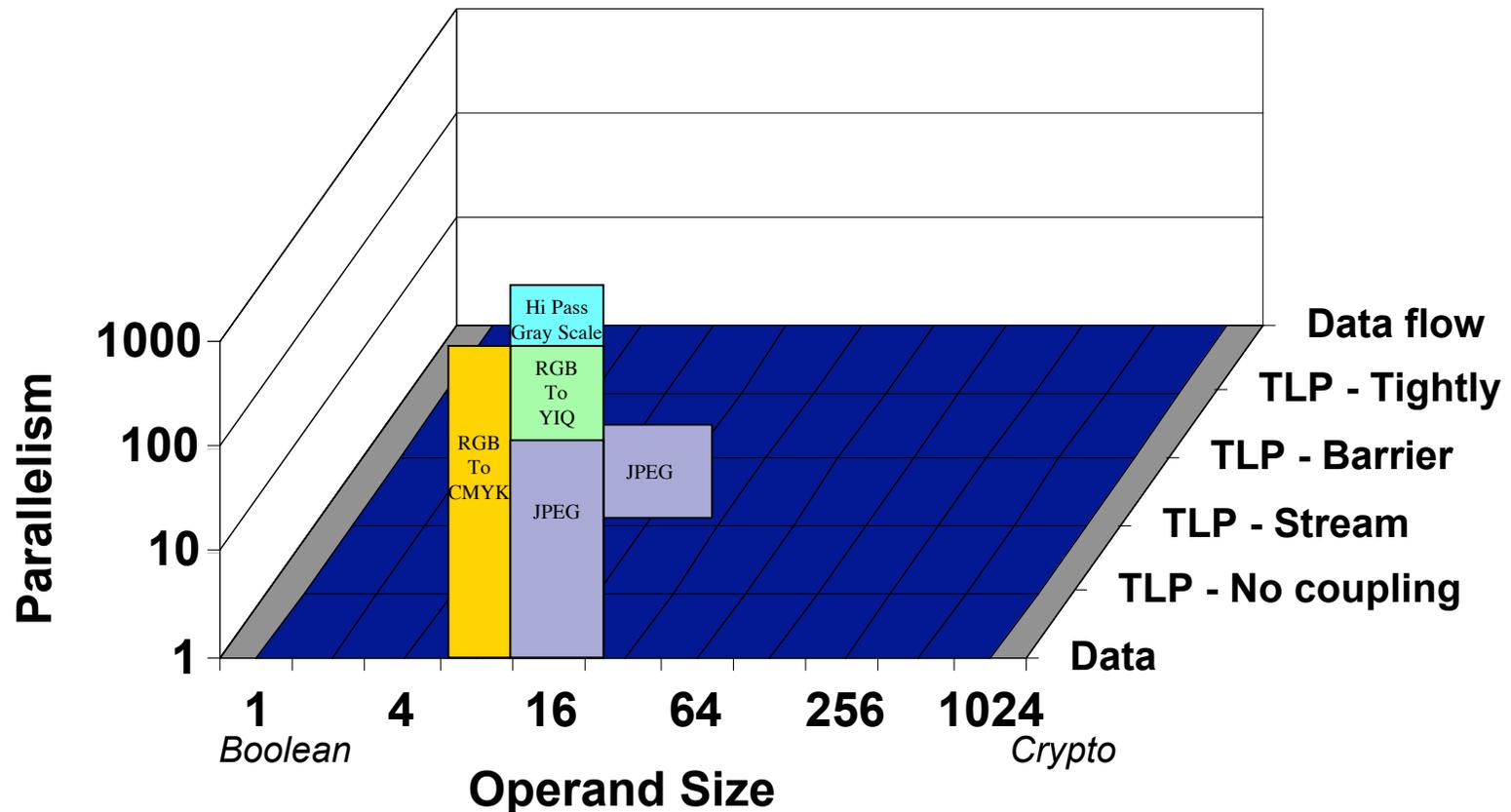
Parallel Framework - Benchmarks

- EEMBC



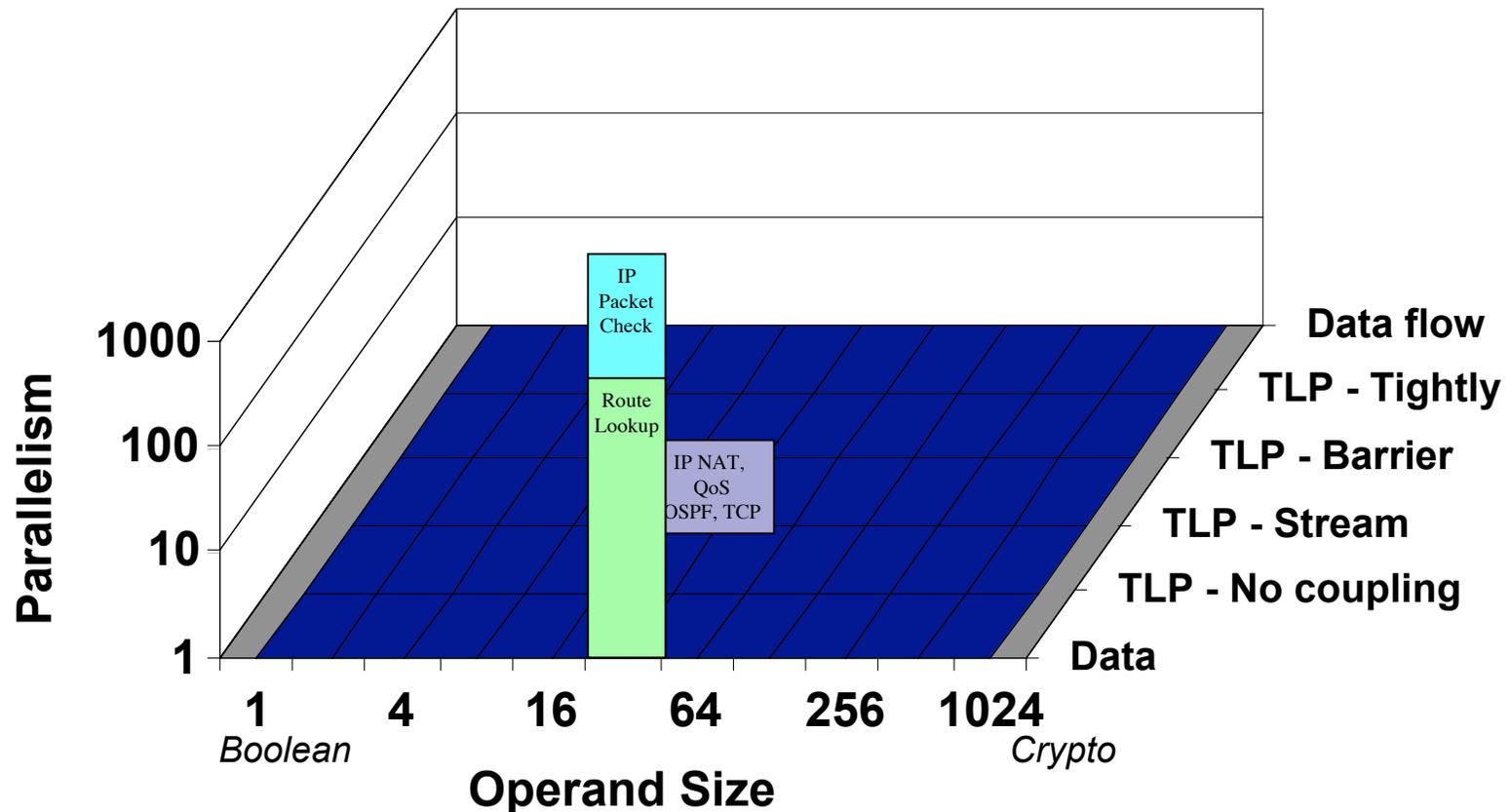
Parallel Framework - Benchmarks

- EEMBC



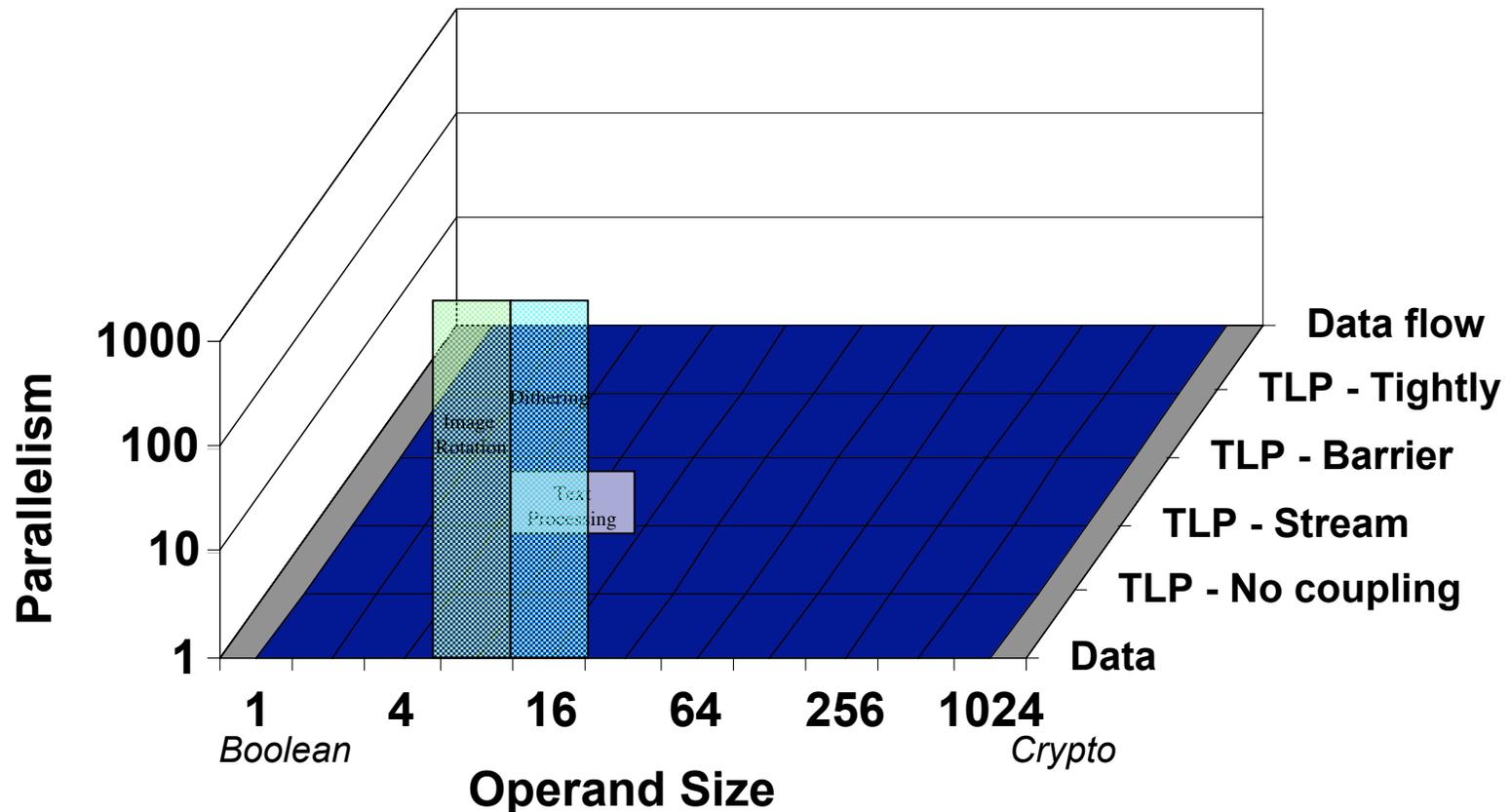
Parallel Framework - Benchmarks

- EEMBC



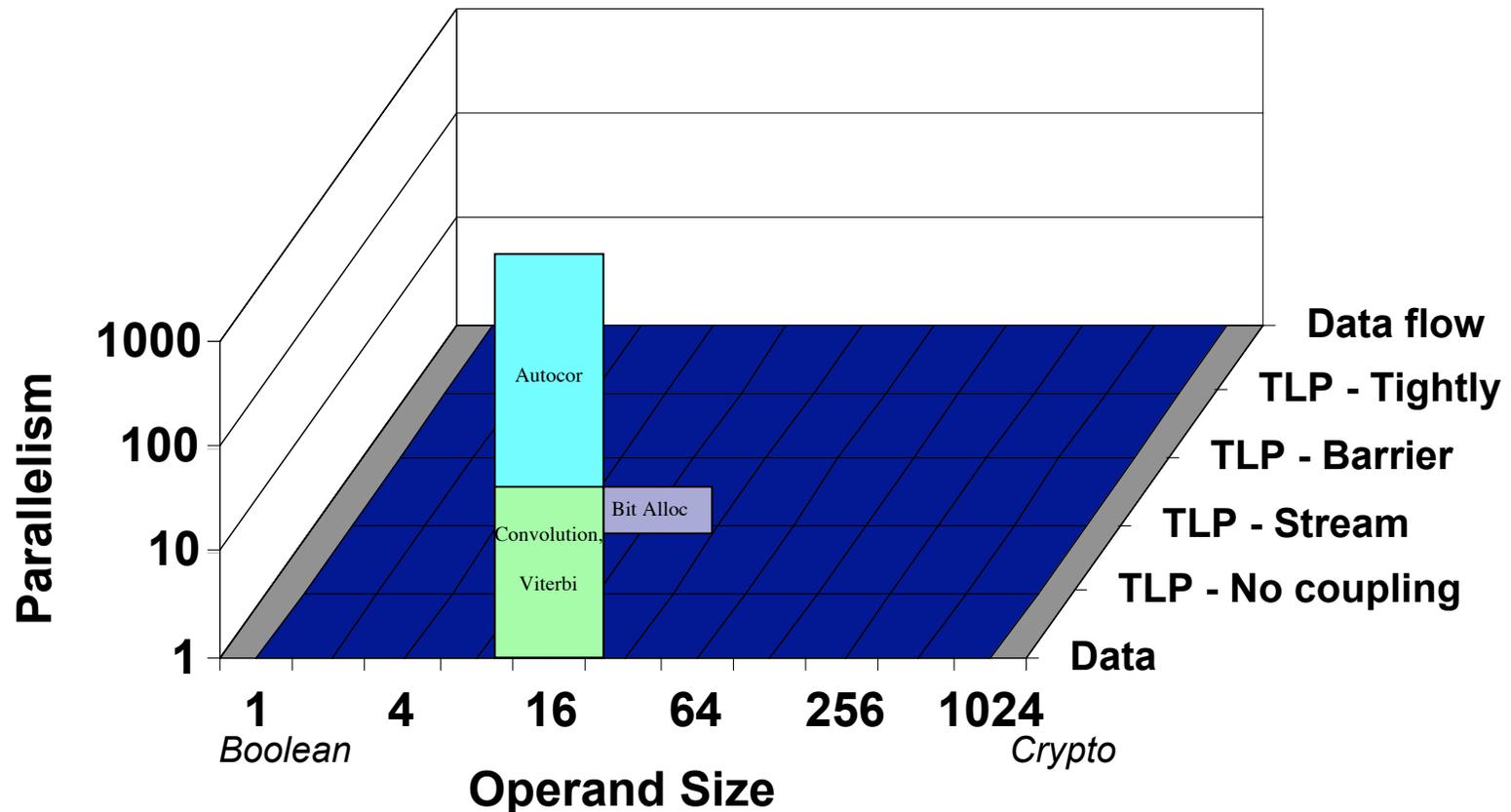
Parallel Framework - Benchmarks

- EEMBC



Parallel Framework - Benchmarks

- EEMBC





SPECintCPU: 32-bit integer

- FSM: perlbench, bzip2, minimum cost flow (MCF), Hidden Markov Models (hmm), video (h264avc), Network discrete event simulation, 2D path finding library (astar), XML Transformation (xalancbmk)
- Sorting/Searching: go (gobmk), chess (sjeng),
- Dense linear algebra: quantum computer (libquantum), video (h264avc)
- TBD: compiler (gcc)

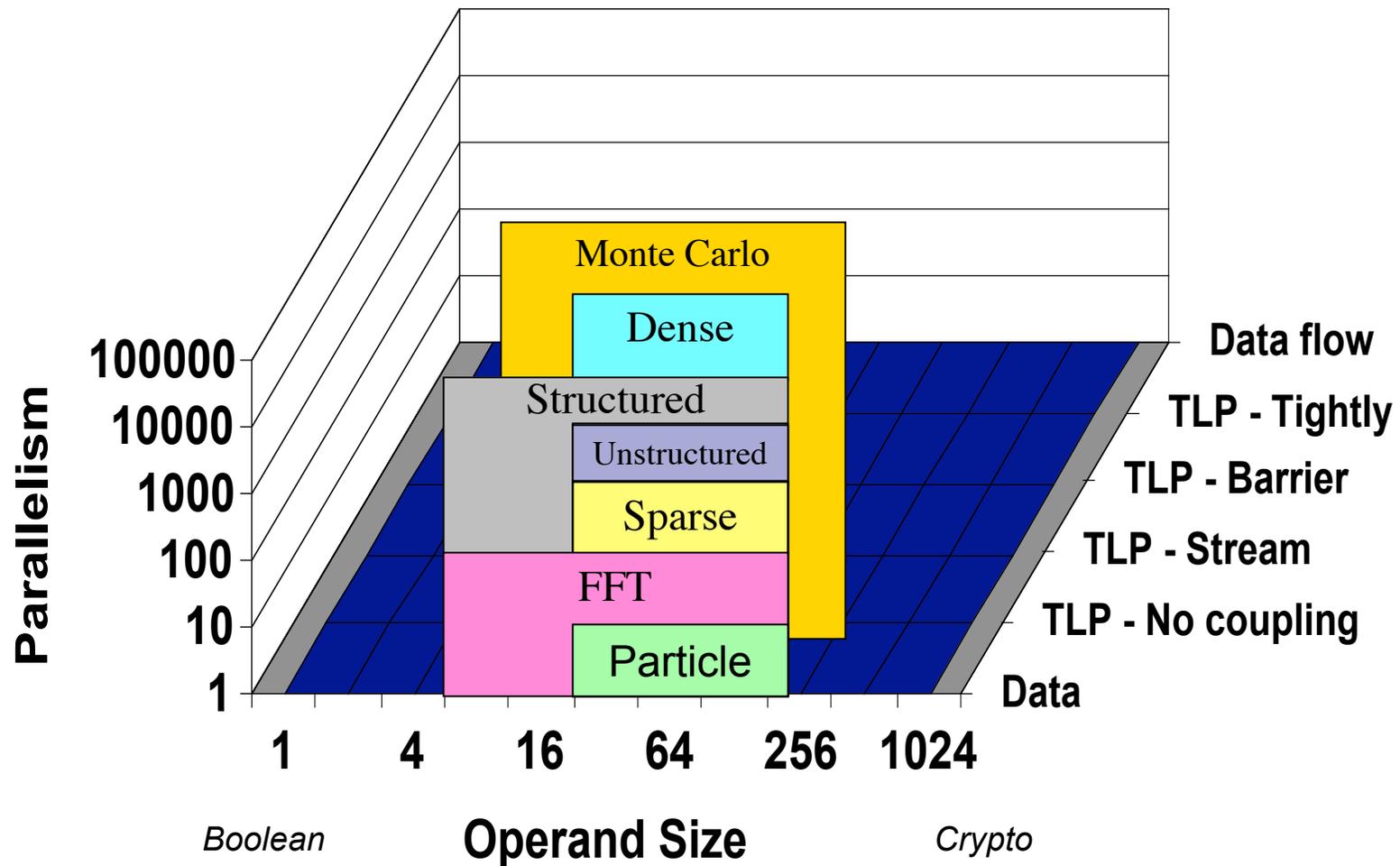


SPECfpCPU: 64-bit Fl. Pt.

- **Structured grid:** Magnetohydrodynamics (zeusmp), General relativity (cactusADM), Finite element code (calculix), Maxwell's E&M eqns solver (GemsFDTD), Fluid dynamics (lbm; leslie3d-AMR), Finite element solver (dealII-AMR), Weather modeling (wrf-AMR)
- **Sparse linear algebra:** Fluid dynamics (bwaves), Linear program solver (soplex),
- **Particle methods:** Molecular dynamics (namd, 64-bit; gromacs, 32-bit),
- **TBD:** Quantum chromodynamics (milc), Quantum chemistry (gamess), Ray tracer (povray), Quantum crystallography (tonto), Speech recognition (sphinx3)

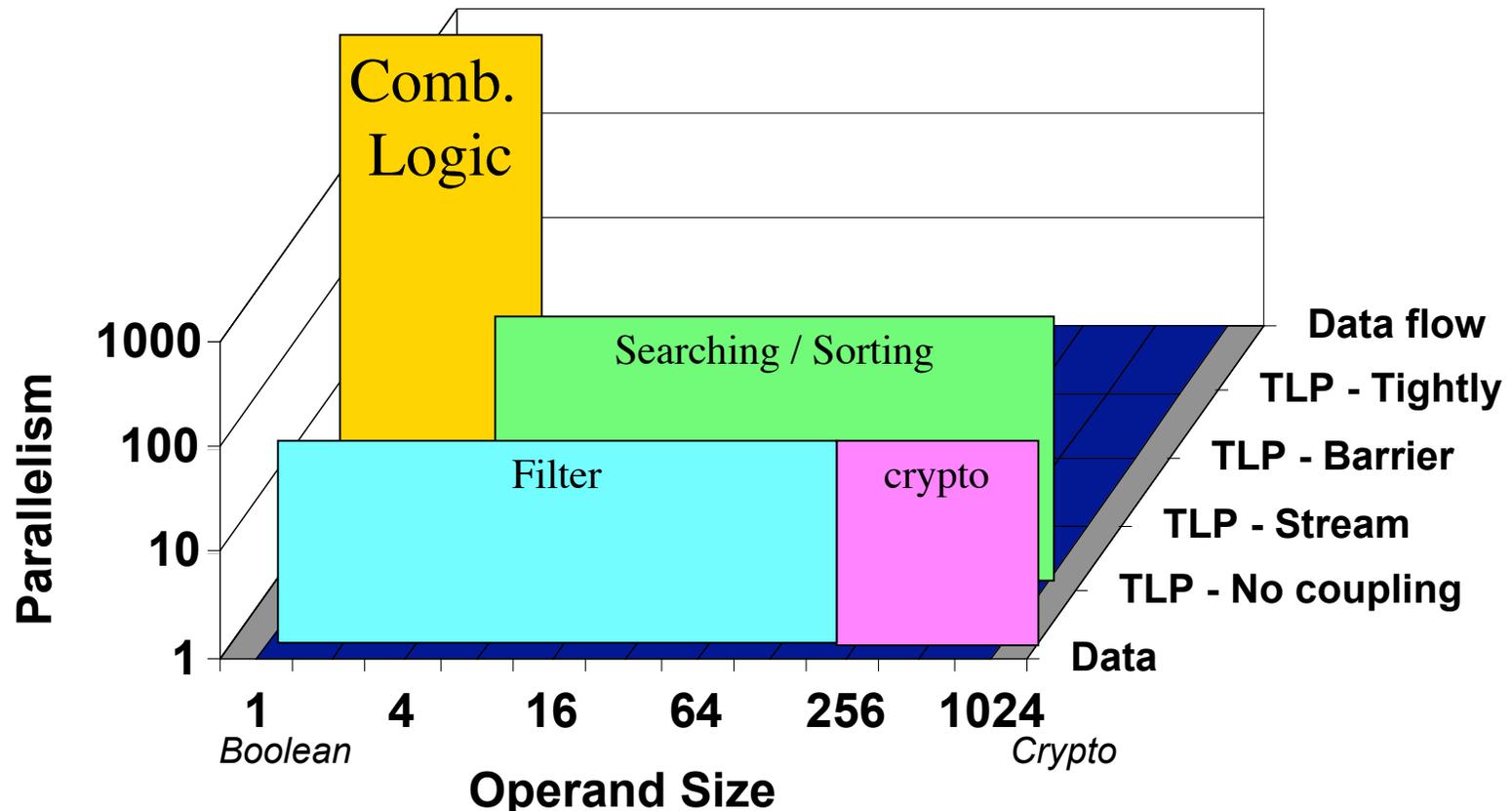
Parallel Framework - Benchmarks

- 7 Dwarfs: Use simplest parallel model that works



Parallel Framework - Benchmarks

- Additional 4 Dwarfs (not including FSM, Ray tracing)



Parallel Framework – EEMBC Benchmarks

<i>Number EEMBC kernels</i>	<i>Parallelism</i>	<i>Style</i>	<i>Operand</i>
14	1000	Data	8 - 32 bit
5	100	Data	8 - 32 bit
10	10	Stream	8 - 32 bit
2	10	Tightly Coupled	8 - 32 bit

