Power Efficiency
by
System Management
and by
System Architecture

Avinoam Kolodny

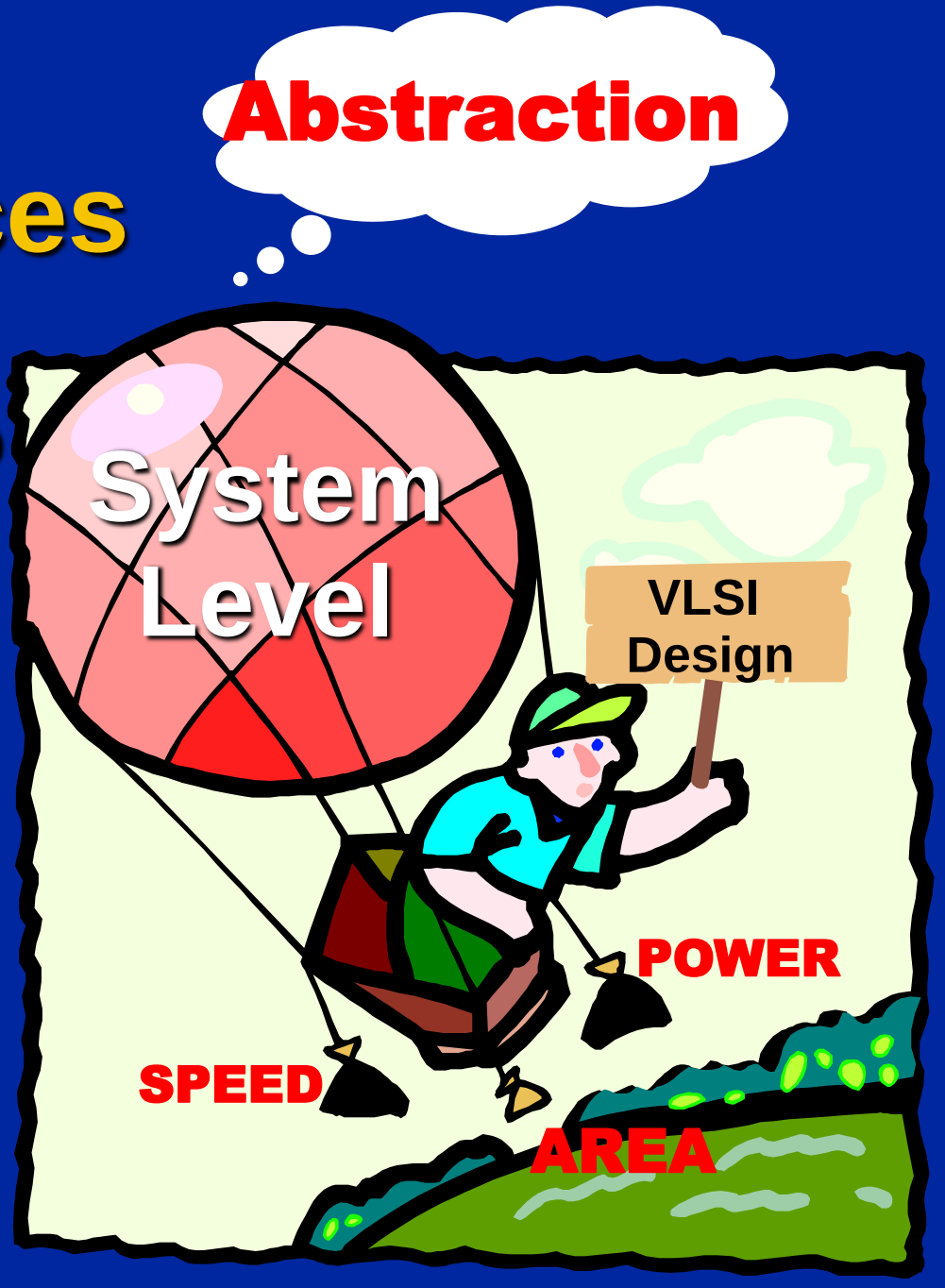
Electrical Engineering Department
Technion – Israel Institute of Technology

Perspective 1:
The Pain Principle

- Design Technology evolves by working on the most painful problem of the day

Perspective 2: Opposing Forces

- Abstraction pulls up
  - Necessary to handle complexity
- Physical issues pull down like gravity
  - Area
  - Speed
  - Power



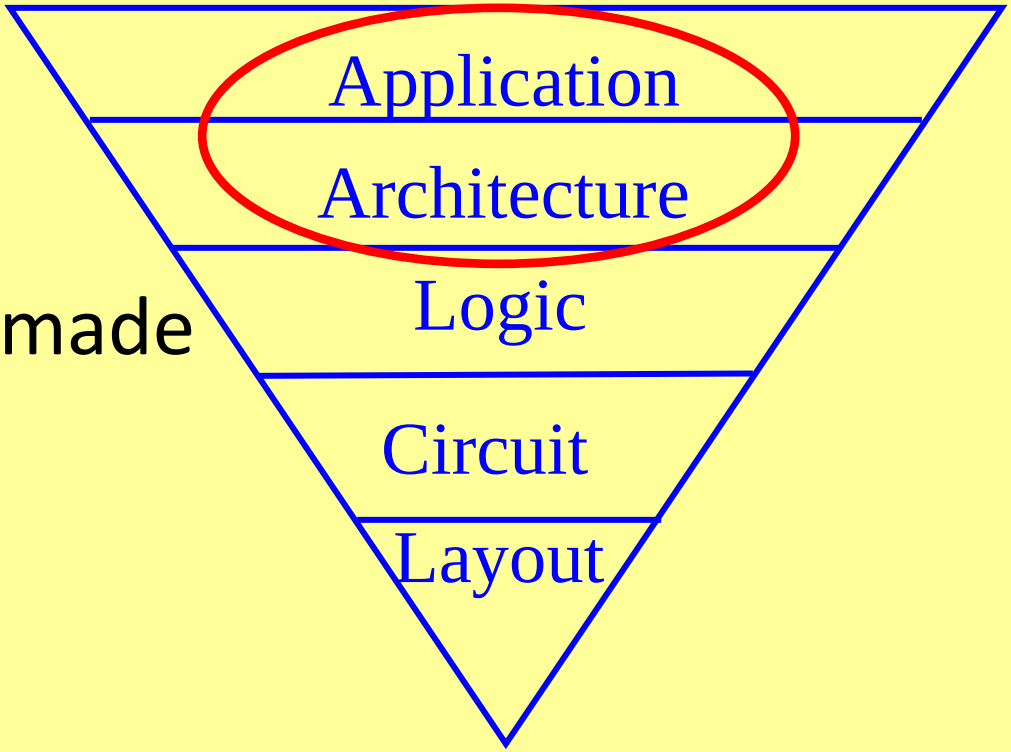
Perspective 3:
The Nature of Low-Power Design

- No critical root cause
  - Power is cumulative

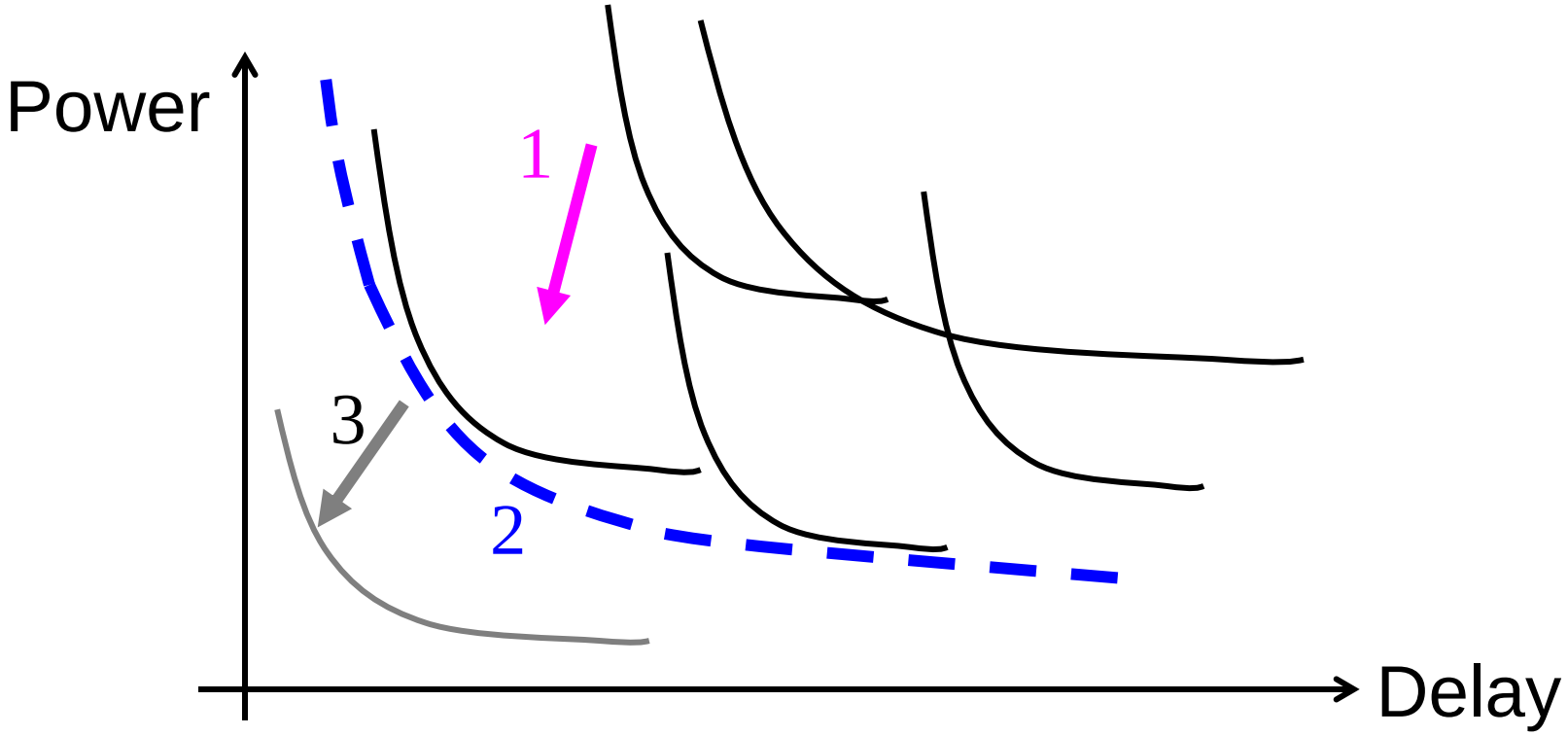
Perspective 3:
The Nature of Low-Power Design

- No critical root cause
  - Power is cumulative

- Most savings can be made at the high levels



Three Types of Action for Low Power

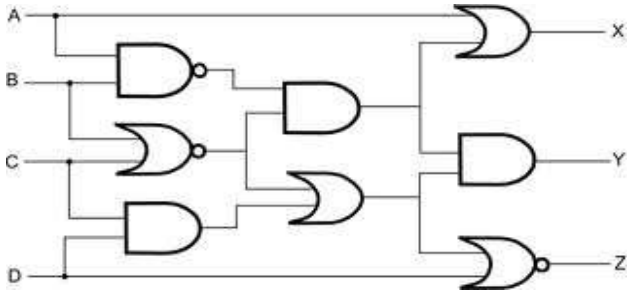


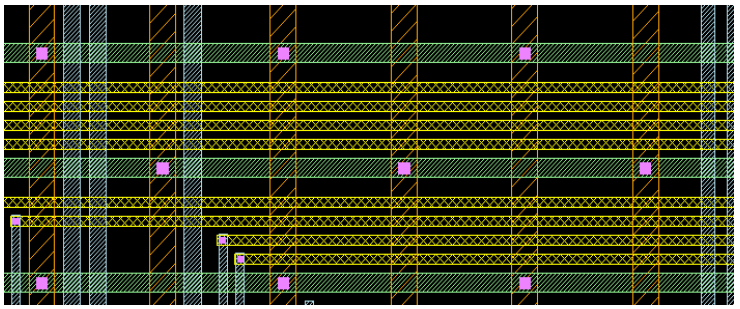
- 1 **Reduce waste of energy** (→system management)
- 2 (Optimal) Tradeoff power with delay
- 3 **Redefine the computational task** (→ system architecture)

* M. Horowitz et al., "Methods for true power minimization," *ICCAD 2002* .

* Y. Aizik and A. Kolodny, *VLSI Design*, 2011.

Changing view of VLSI systems





“Old” view:

- **Speed and power are dominated by ALU operations**
- **Communication is immediate**

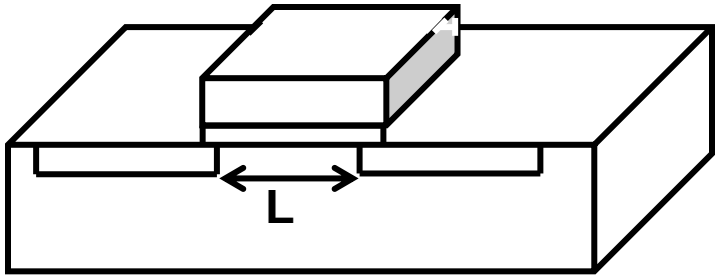
“New” view:

- **Speed and power are dominated by communication**
- **Computing operations are fast and cheap**

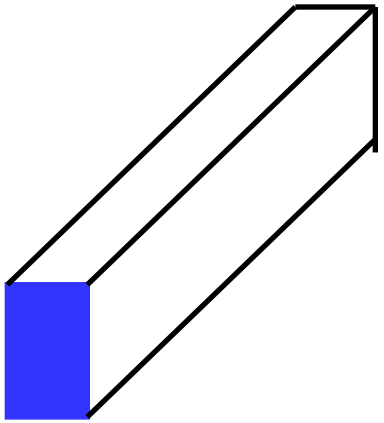
The truth is actually somewhere in the middle…
… that’s why architecture is challenging!

Why Views Are Changing?
(Technlology Scaling)

- **For transistors, smaller is better!**
  - Speed
  - Lower power
  - Lower cost

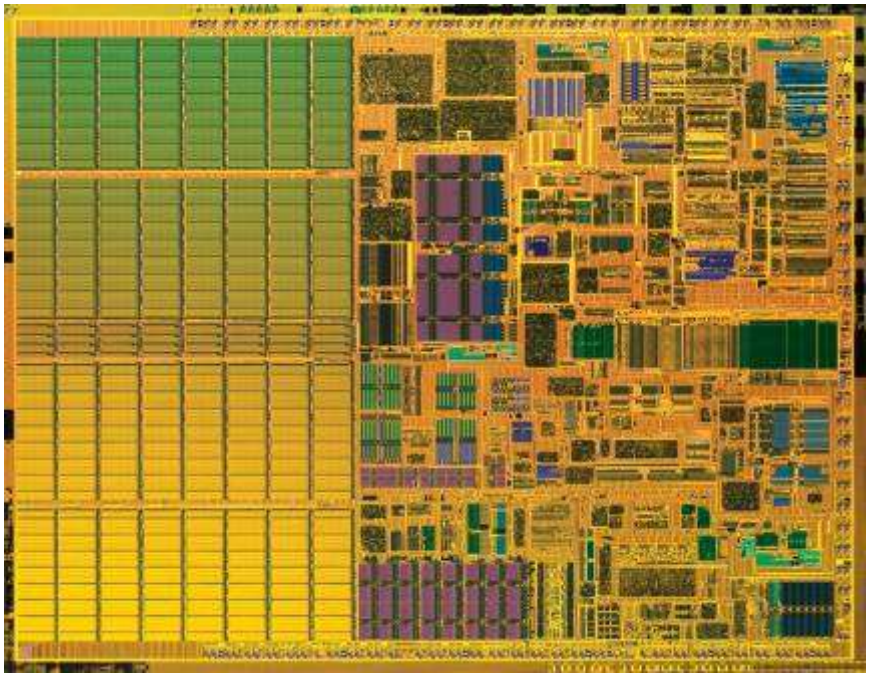


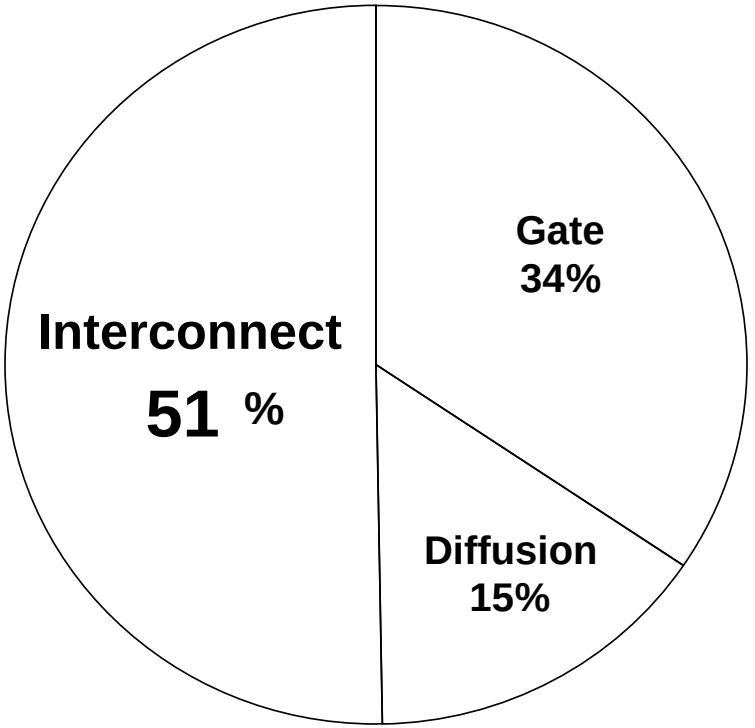
- **For wires, smaller is worse…**



A case study - 2004

- **Intel's Pentium-M, low-power microprocessor designed in Haifa, 0.13 micron CMOS**
- **Bit-Transportation energy is larger than computation energy!!!**



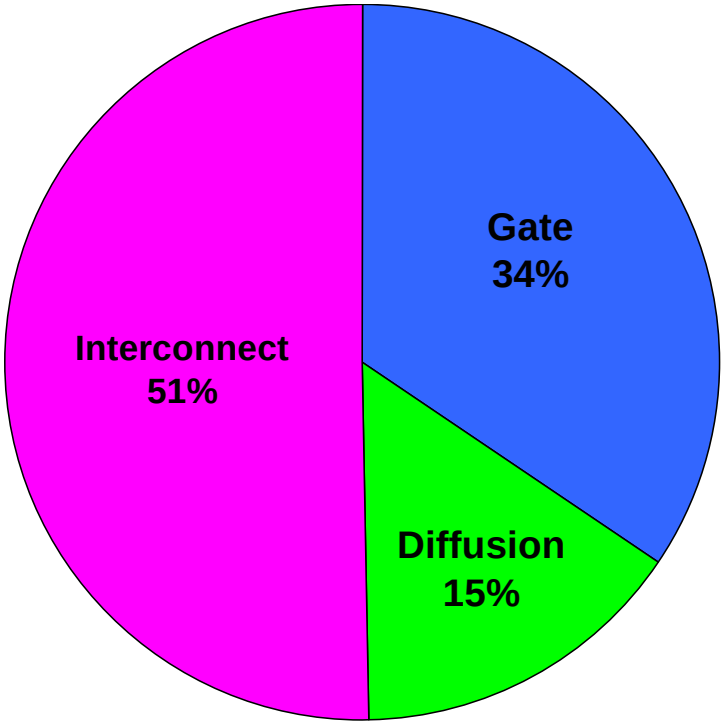


* N. Magen, A. Kolodny, U. Weiser and N. Shamir, "Interconnect-Related Energy dissipation in a Low-Power Microprocessor", Proc. SLIP, 2004.

On-Chip Interconnect Bottleneck

Interconnect Delay is dominant

Interconnect Power is dominant

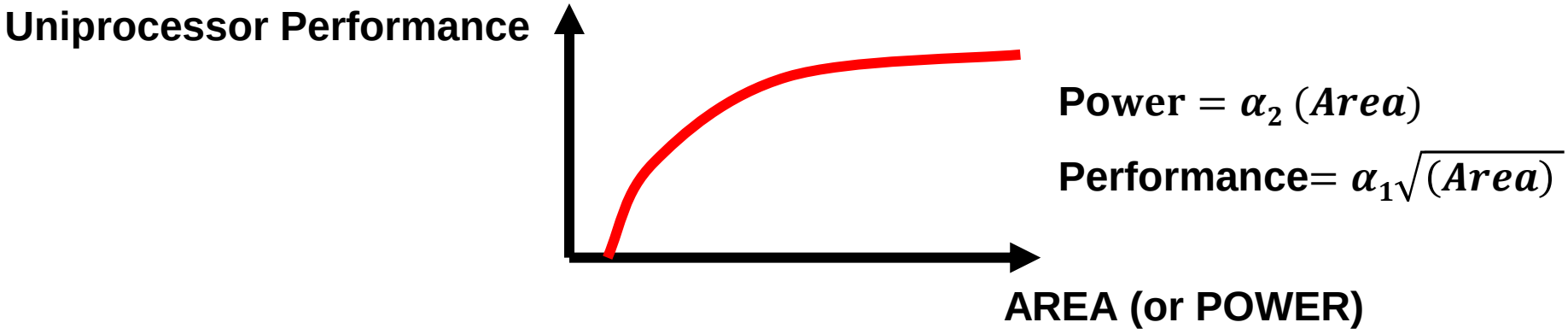


Source: Bohr, IEDM 1995

Source: Nir Magen et al., SLIP 2004.
(Data for Intel "Banias" centrino processor)

If Bits were Cars….

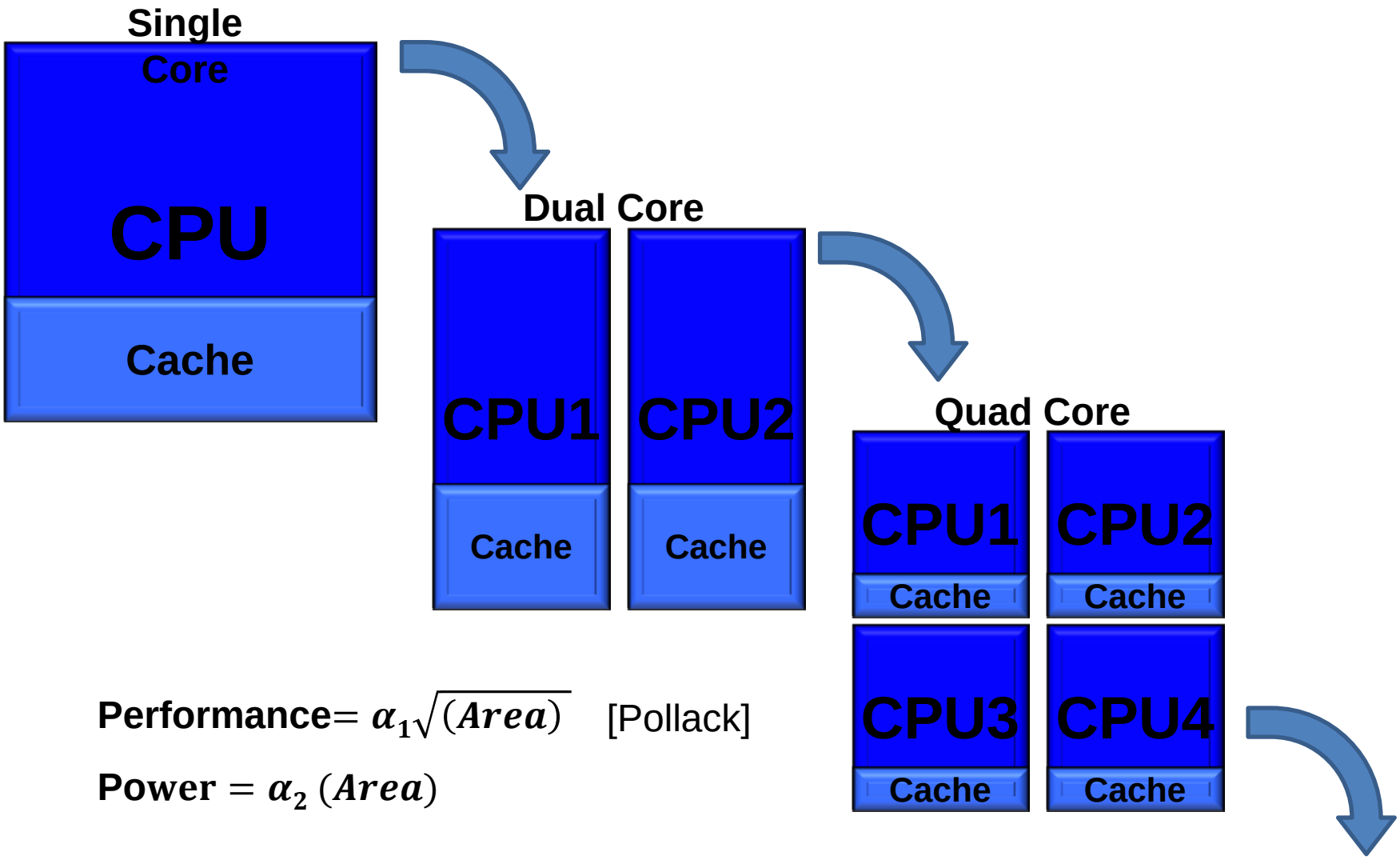
Fred Pollack's Rule



- Power-efficiency requires many parallel local computations
  - Chip Multi Processors (CMP)
  - Thread-Level Parallelism (TLP)

* F. Pollack, MICRO 1999.
* S. Borkar, DAC 2007.

Processor System Evolution



Performance$= \alpha_1 \sqrt{(Area)}$ [Pollack]

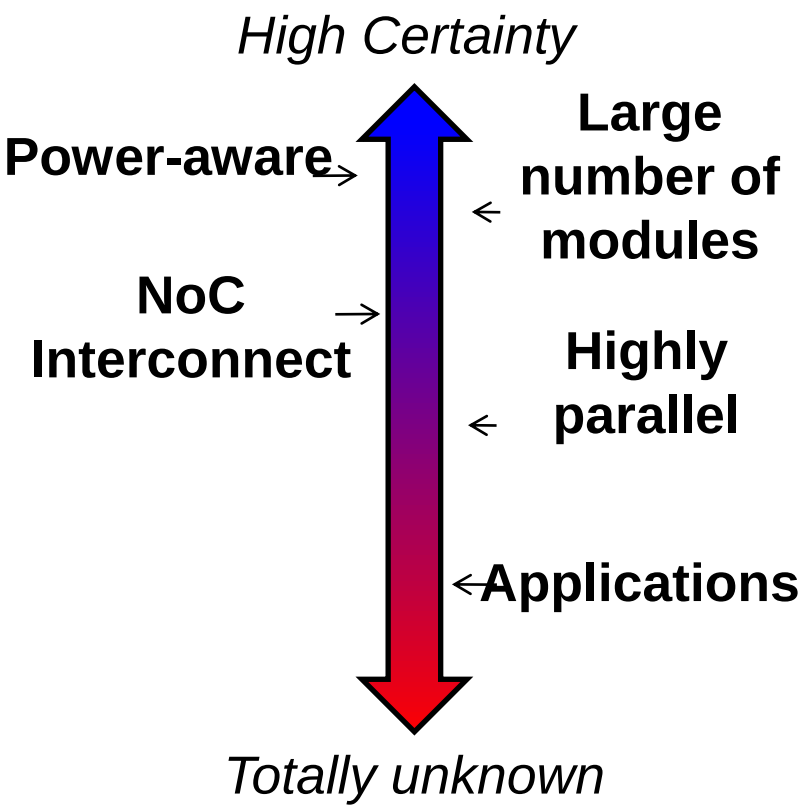
Power $= \alpha_2 \, (Area)$

Multi-Core as a ‘Hail Mary’

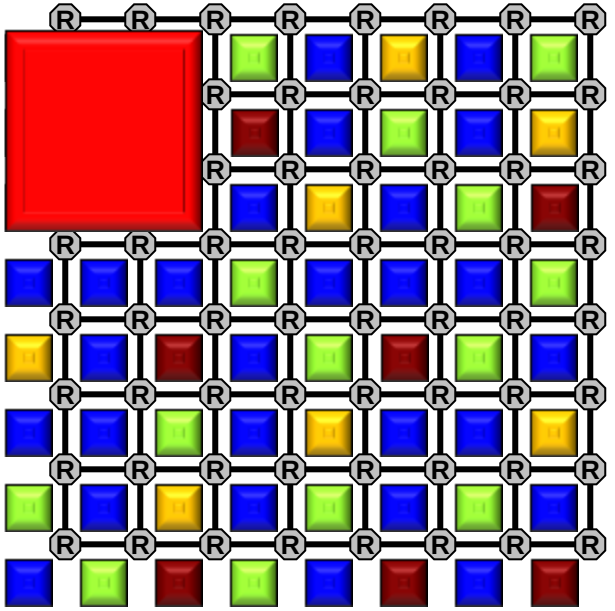
“….. the semiconductor industry threw the equivalent of a Hail Mary pass when it switched from making microprocessors run faster to putting more of them on a chip—doing so without any clear notion of how such devices would in general be programmed. The hope is that someone will be able to figure out how to do that, but at the moment, the ball is still in the air.”

* Prof. David Patterson, *IEEE Spectrum*, July 2010

What do we know about future systems?



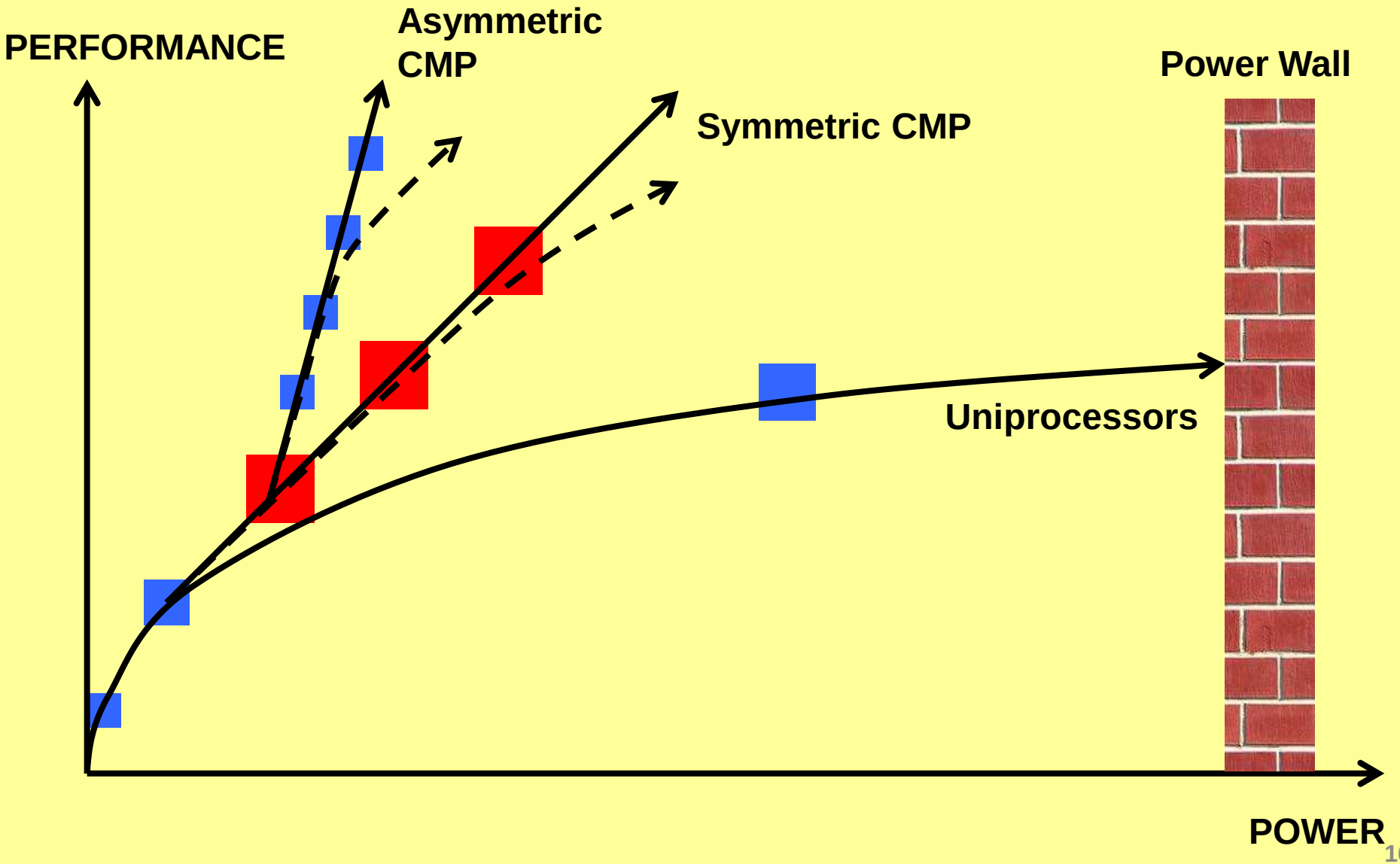
"Asymmetric"



- Highly regular
- Classes of Replicated cores
  - Standard modules (DSP, HW accelerators, Cache banks, etc.)
- Network on Chip
- Power management
  - Different clocks
  - Different operating voltages

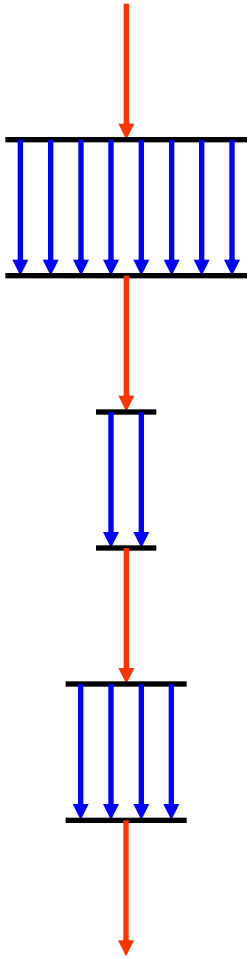
* I. Walter et al., NoCArc 2010.

Processor Architectures:
Uni-core, symmetric multicore, Asymmetric



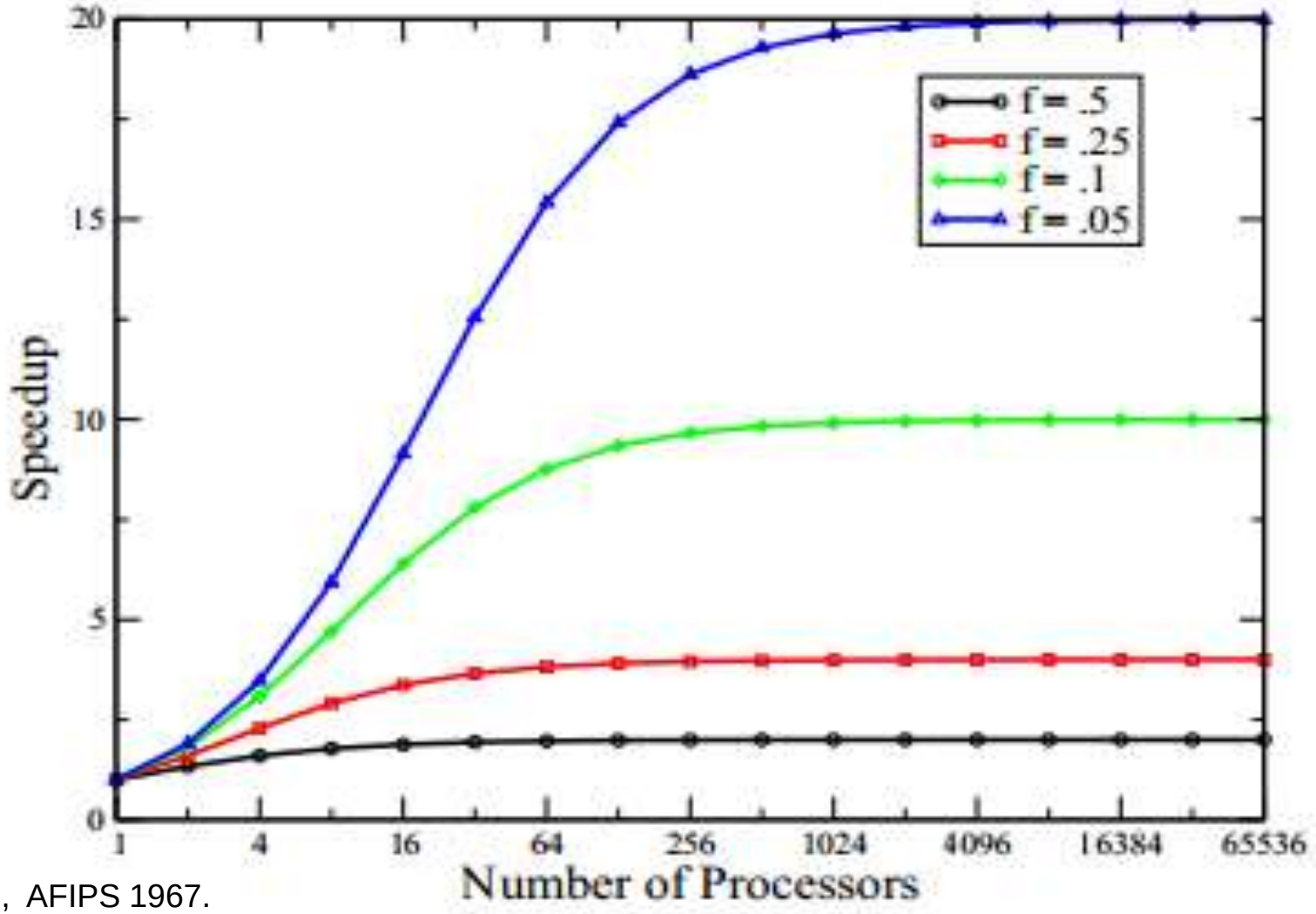
Serial and Parallel Phases in Programs

- Multithreaded programs have serial as well as parallel phases



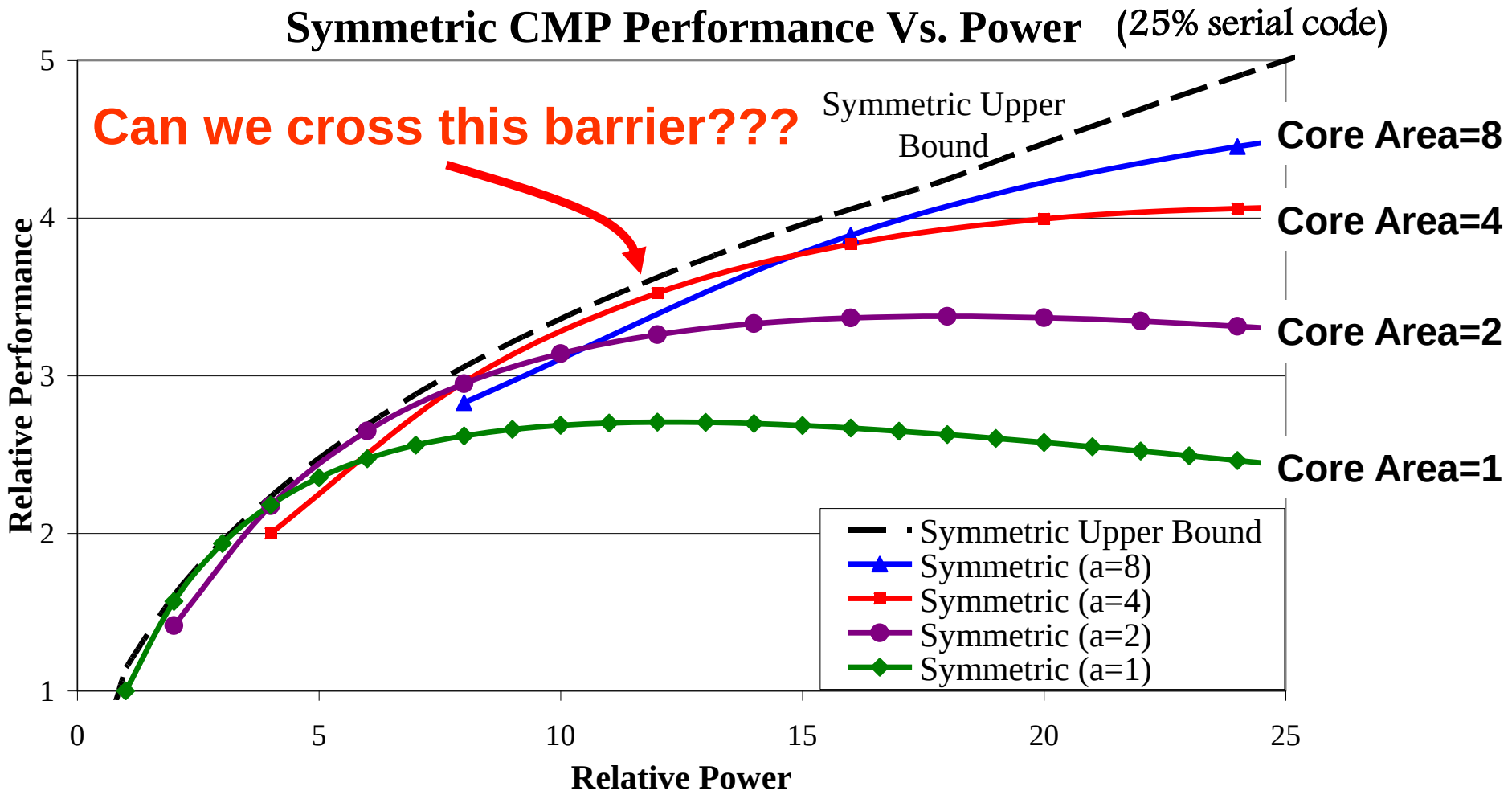
Amdahl's Law

- Performance speedup is limited by serial code
- Speedup is limited by 1/f (f=fraction of serial code)



* G. Amdahl, AFIPS 1967.

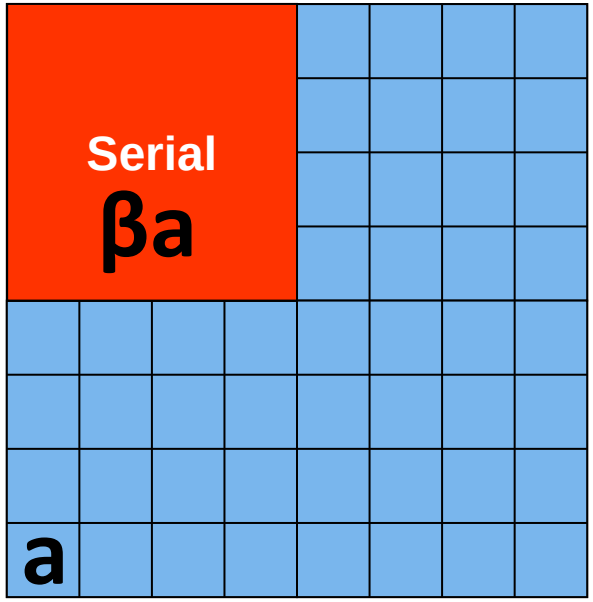
Symmetric Multi-Core Performance Model
(Including interaction overheads)

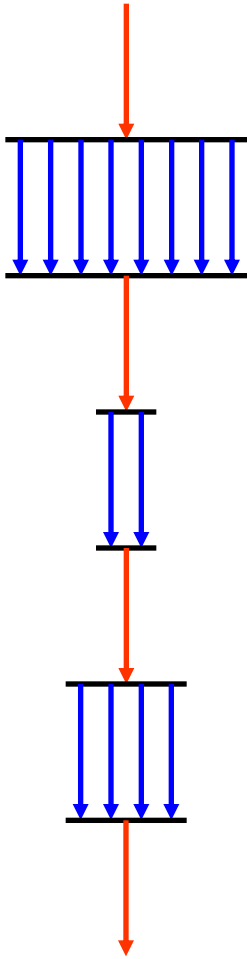


* T. Morad, U. Weiser, A. Kolodny, M. Valero and E. Ayguade,
"Performance, Power Efficiency and Scalability of Asymmetric Cluster Chip Multiprocessors,"
IEEE Computer Architecture Letters, vol. 4, 2005.

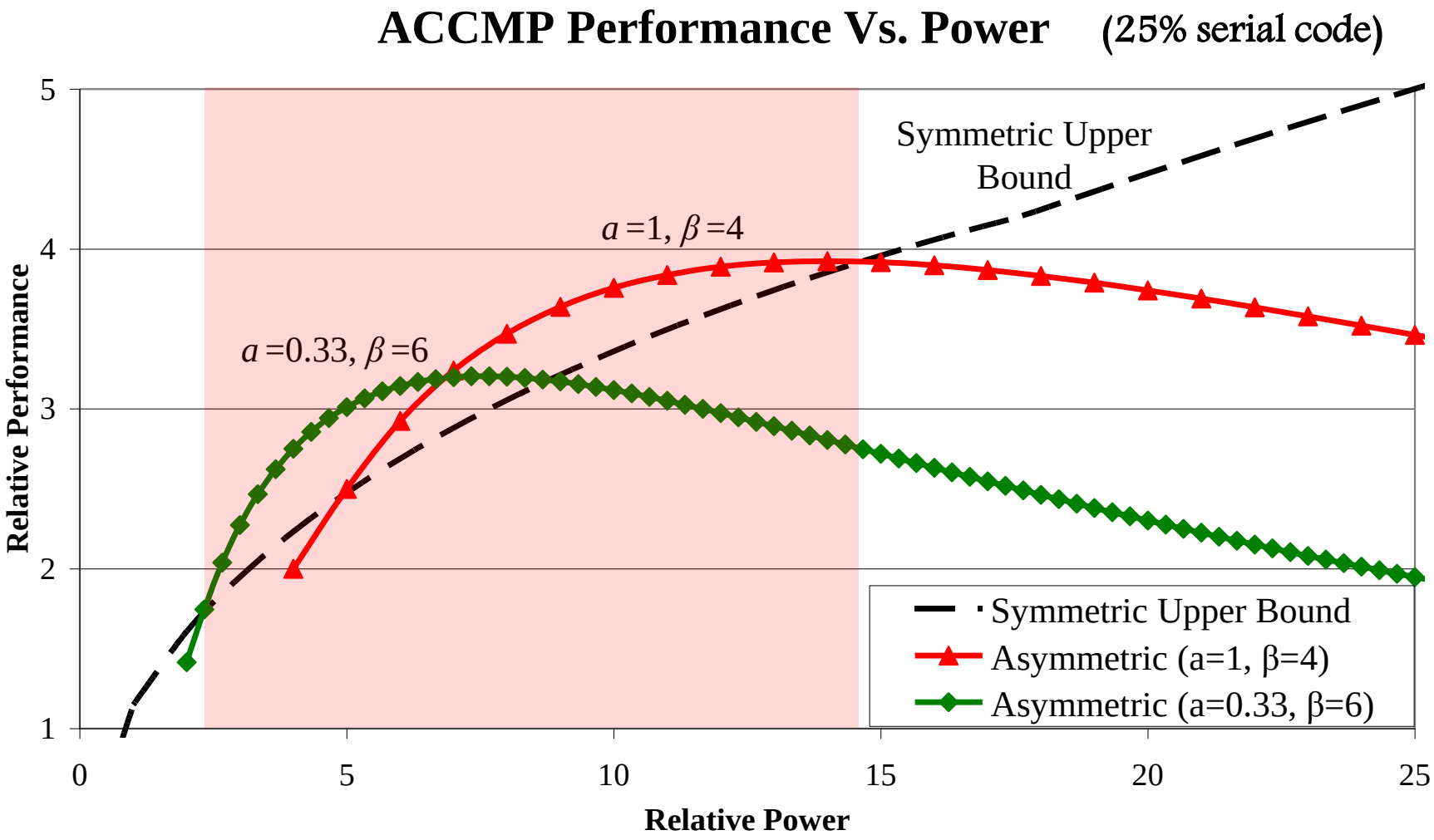
Asymmetric (=Heterogeneous) **Multi-Core**

- Large core area: βa – used for serial code
- Small cores of area: a
- Parallel phases execute on all cores



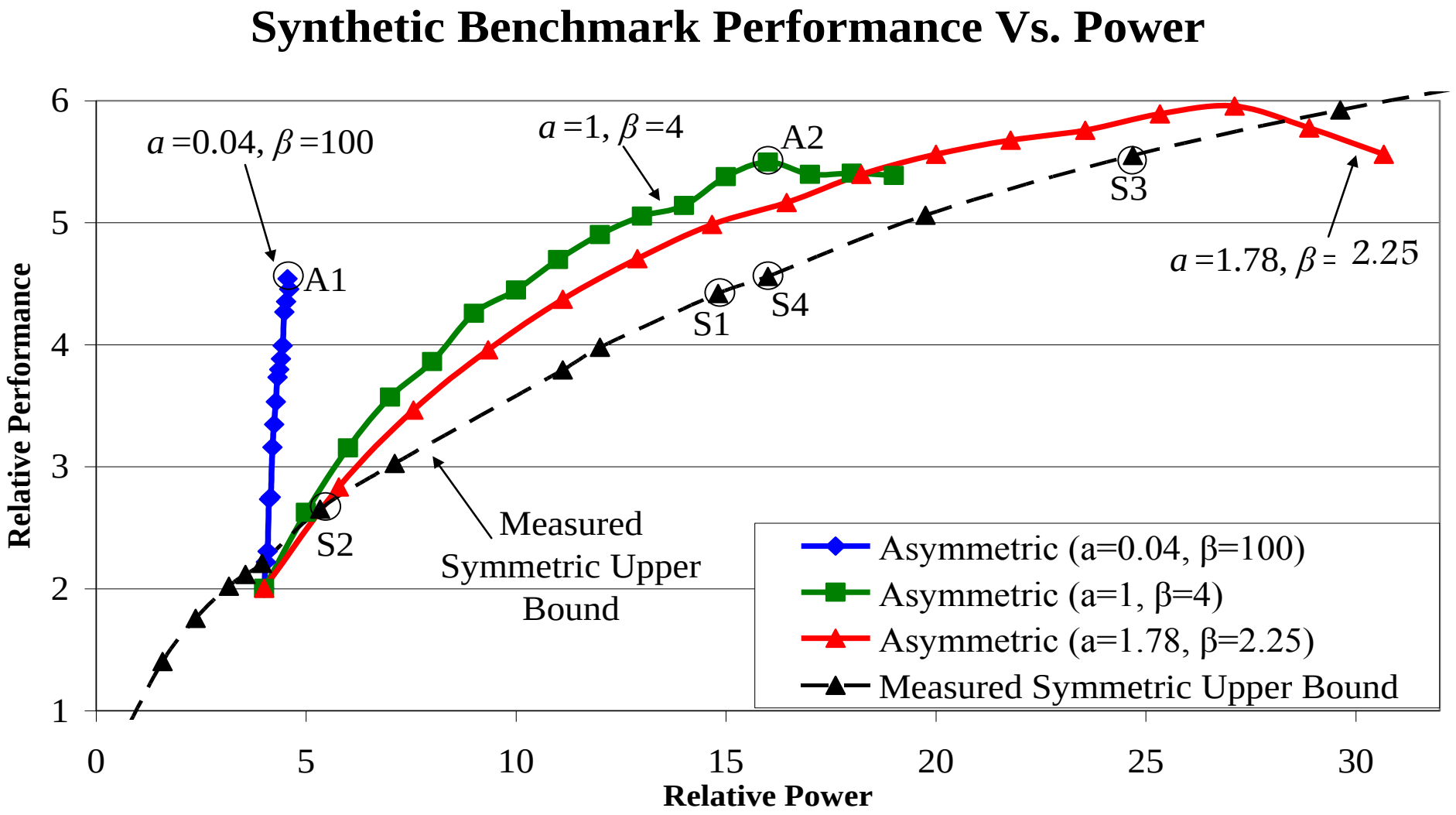


Asymmetric Multi-Core Performance

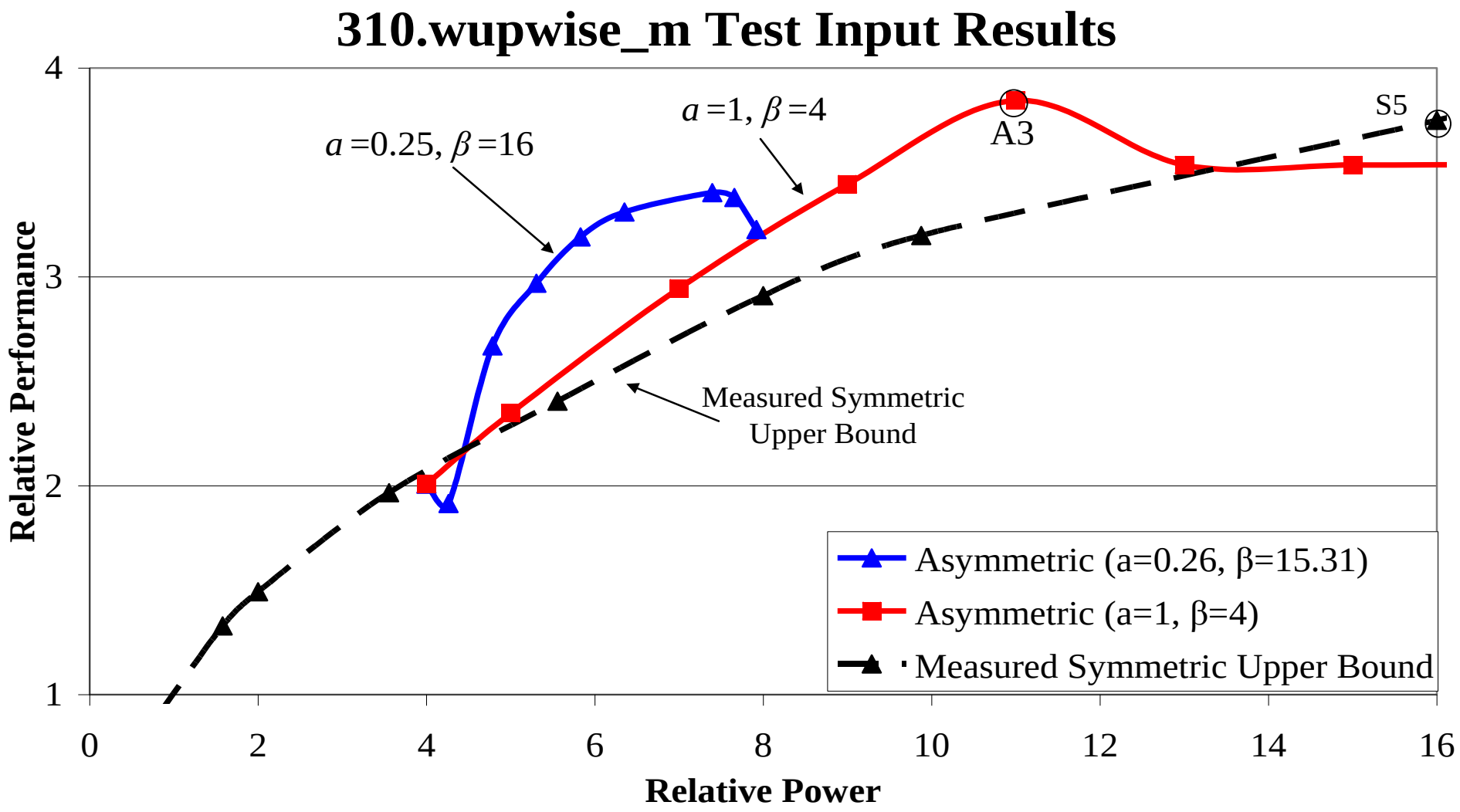


* T. Morad, U. Weiser, A. Kolodny, M. Valero and E. Ayguade, IEEE Computer Architecture Letters, vol. 4, 2005.
* M. Hill and R. Marty, *Computer*, July 2008.

Synthetic Benchmark (Emulation)

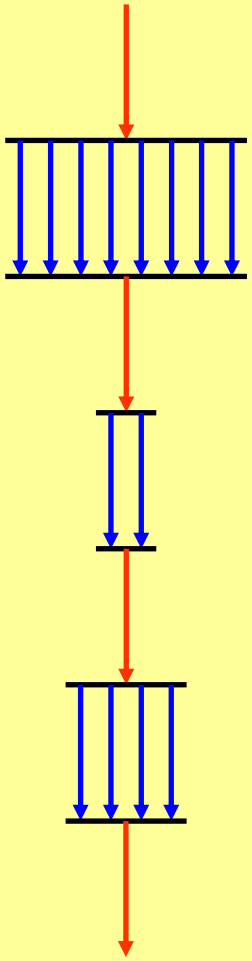


310.wupwise (SPEC-OMP test)

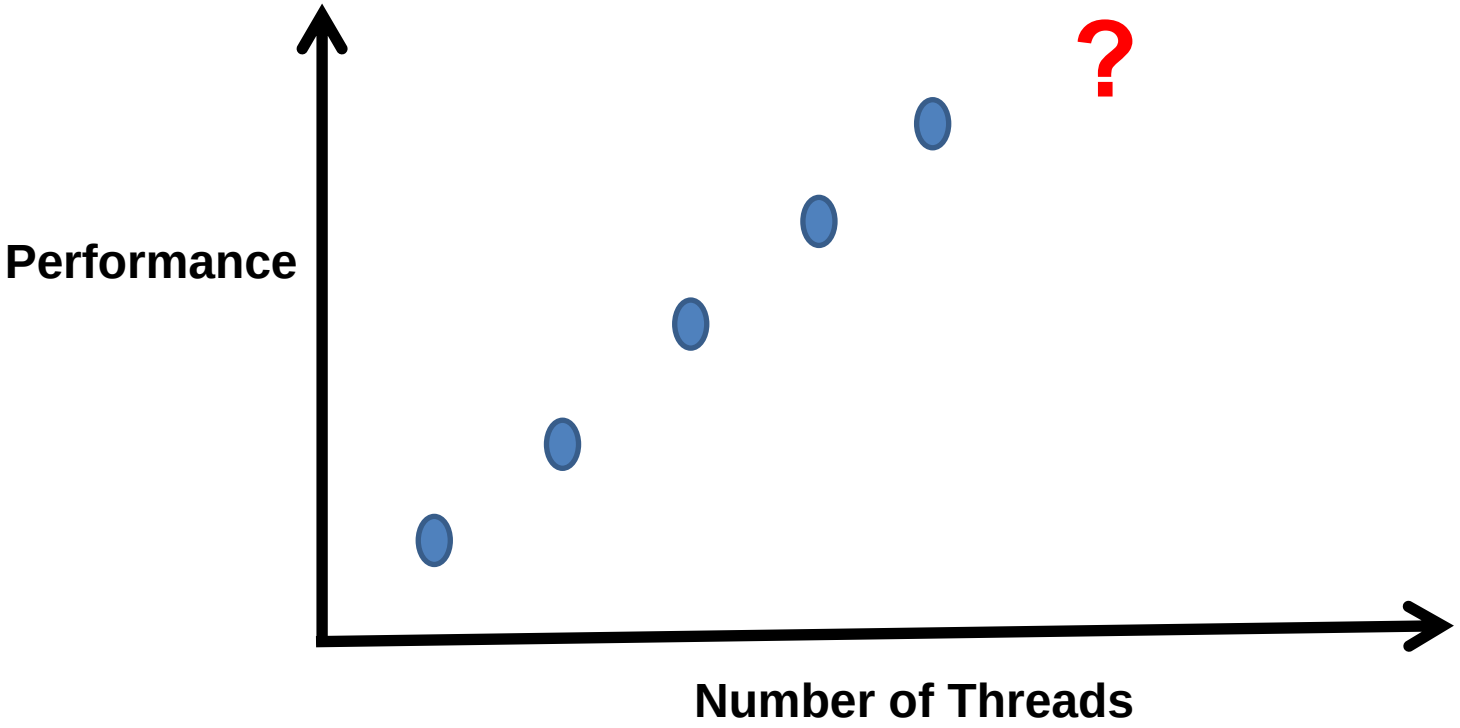


So:
Asymmetric Architecture Can Improve Power Efficiency

How Much Parallelism Exists in Real Programs?



A Useful Plot for Multi-Threaded Systems

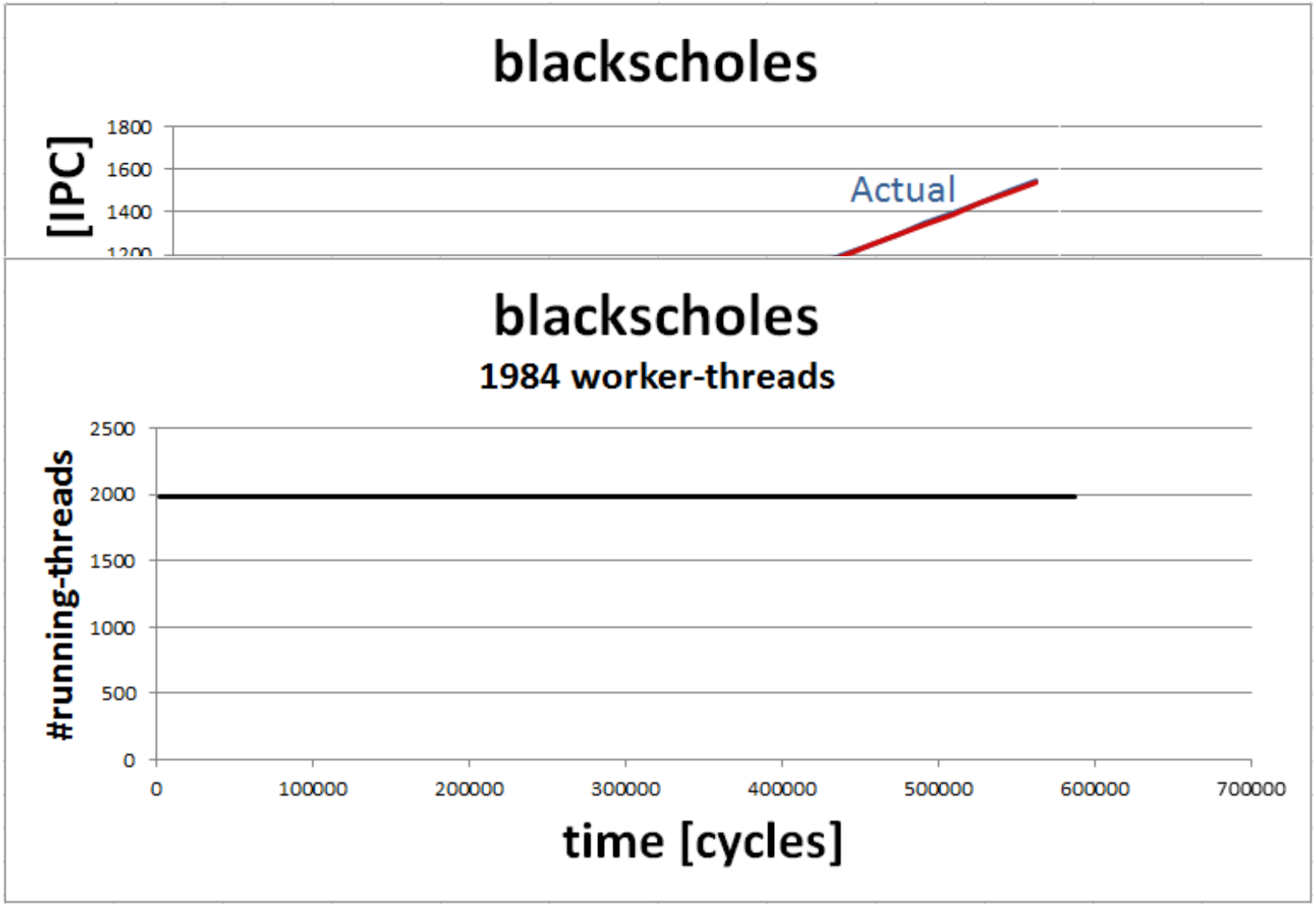


Performance scalability study

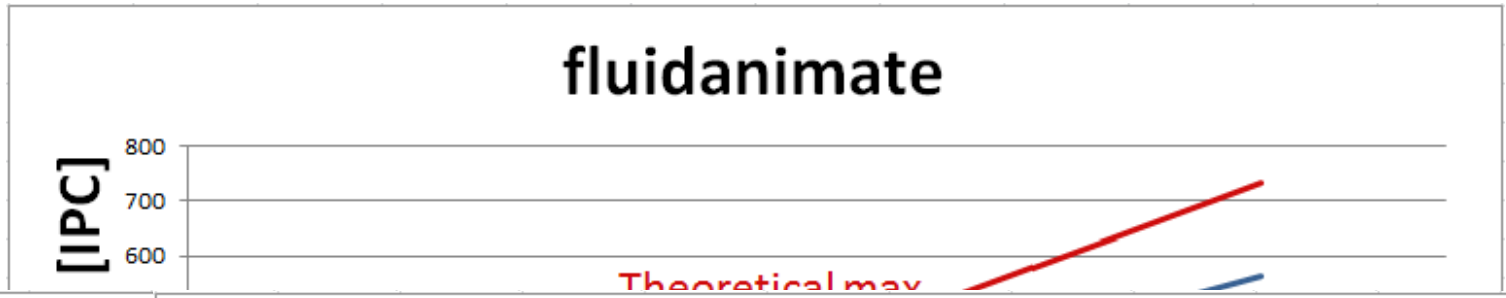
- Use PARSEC
- Capture the parallelism limitation of the **algorithm**
- Use architecture model with no constraints
  - No shared resources – cache, bandwidth
- Perfect memory system – 1 cycle latency
- Parallelism limiting factor:
  inter-thread synchronization

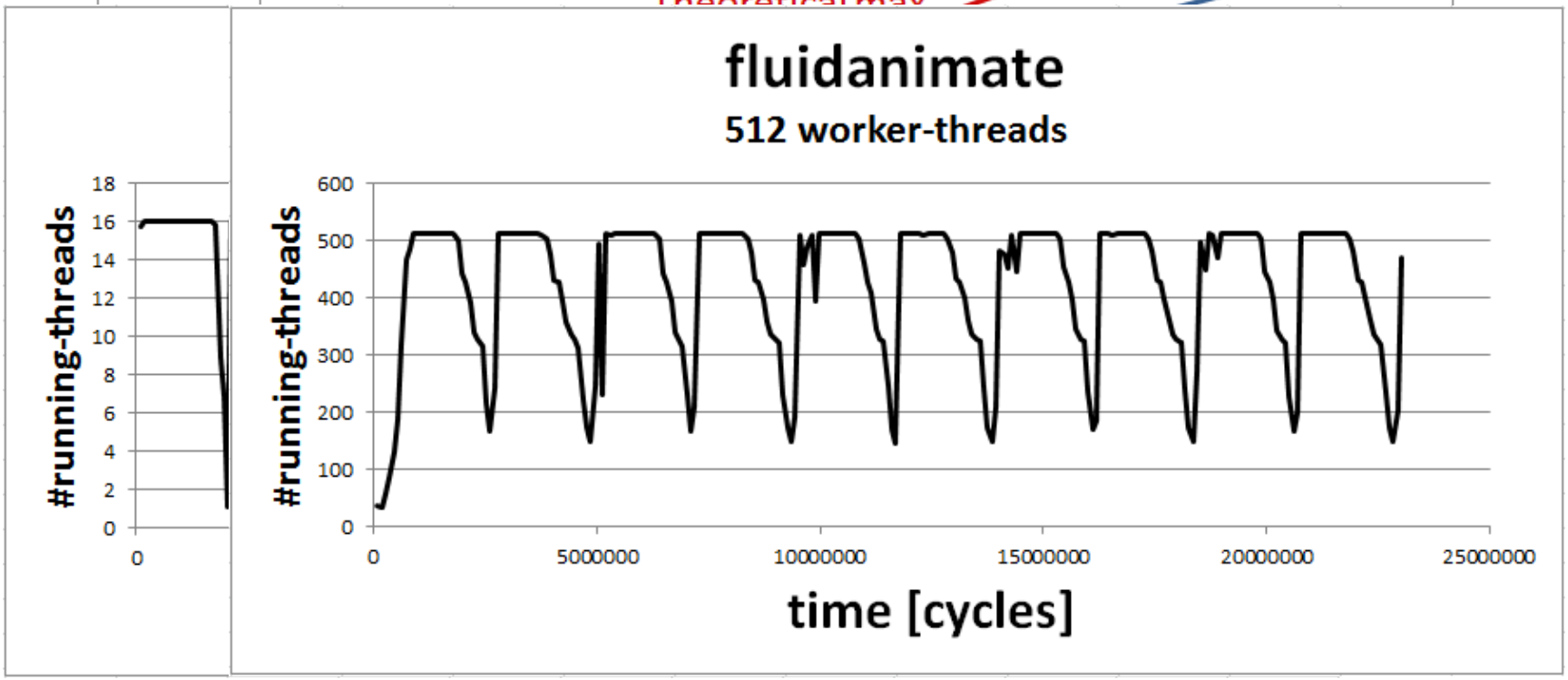
* O. Itzhak M.Sc. thesis., 2013.

Perfect parallelism scalability: blackscholes

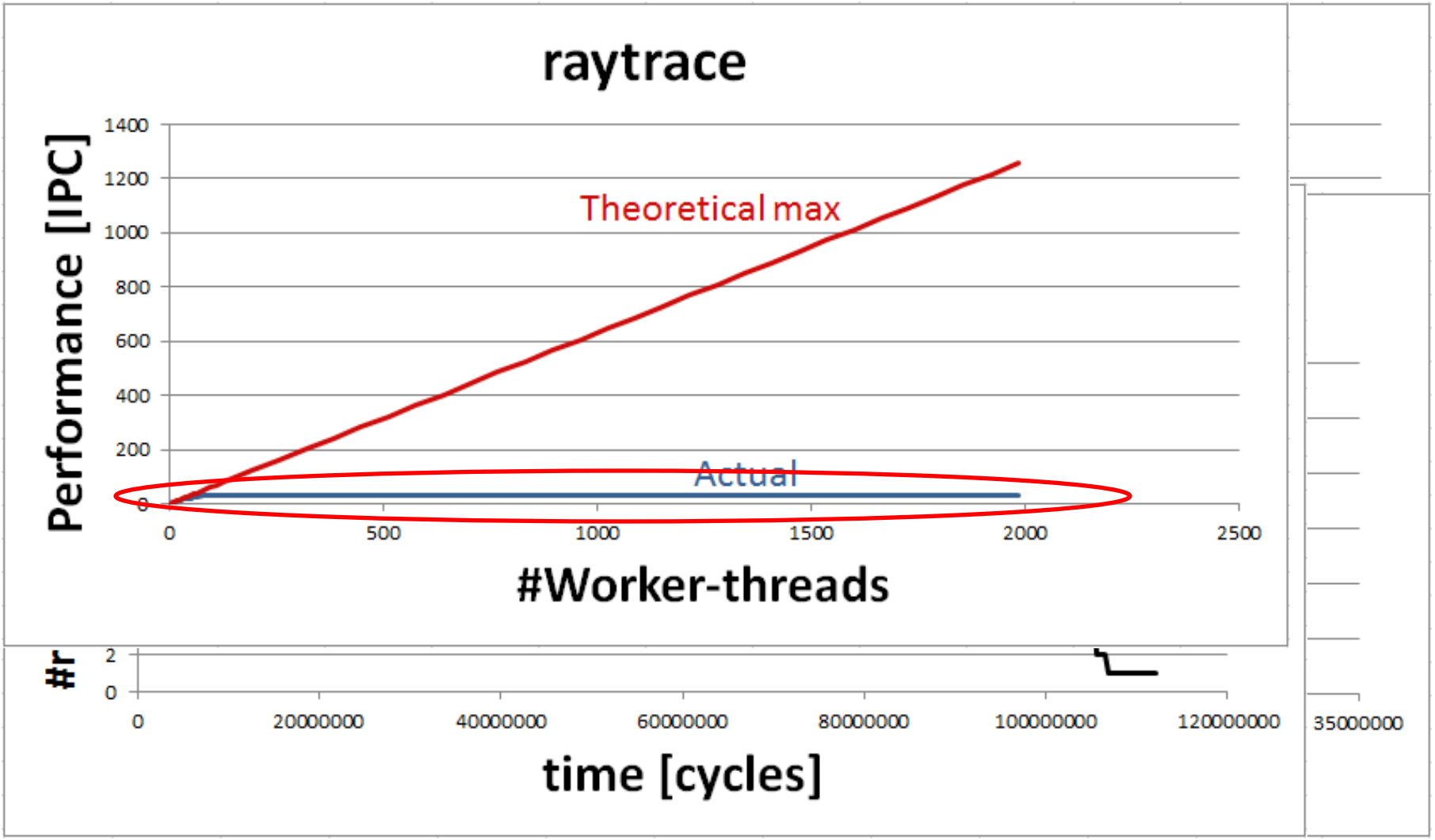


Good parallelism scalability: fluidanimate



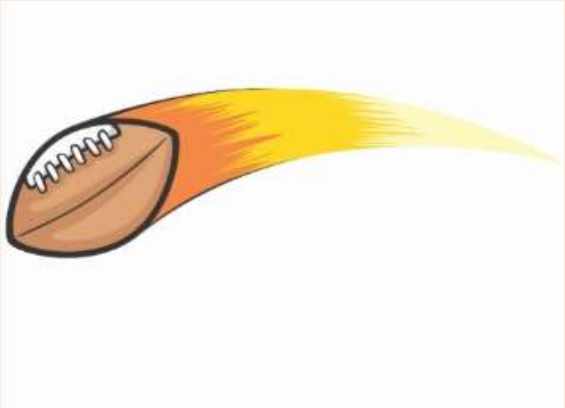


Poor parallelism scalability raytrace

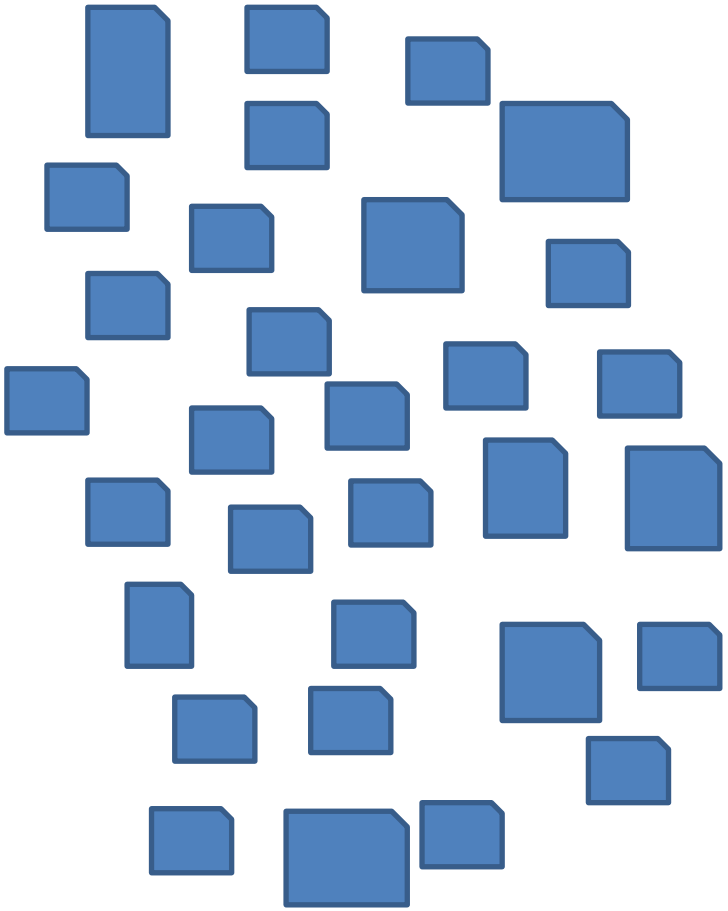


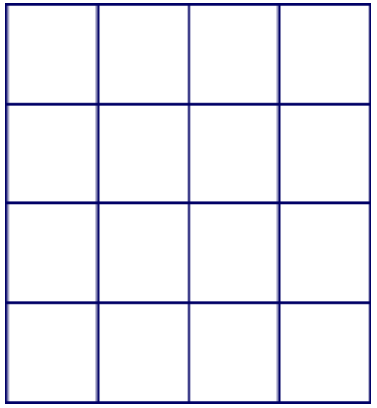
Intermediate Conclusion:

Many multi-threaded applications are not scalable: **They cannot exploit a large number of cores.**

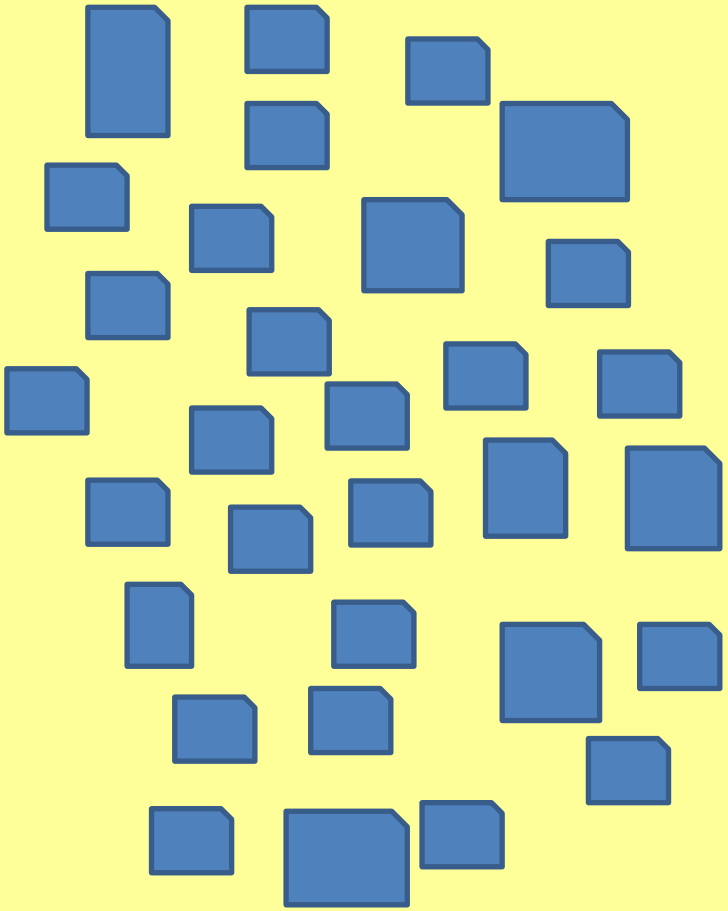


Typical System

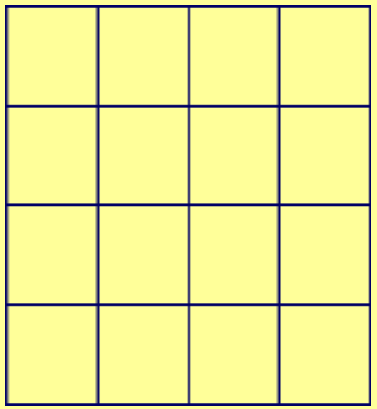




Saving Power by System Management:
Symmetric System, Unrelated Threads



Should we try to maximize utilization of all cores?



Accounting for Energy Spent by Cores in a System

- Energy spent while doing useful work
- Energy spent while waiting for work
- Energy spent on unnecessary or redundant work

Wasteful Effects of Shared Resources (↑ Energy, ↓ Throughput)

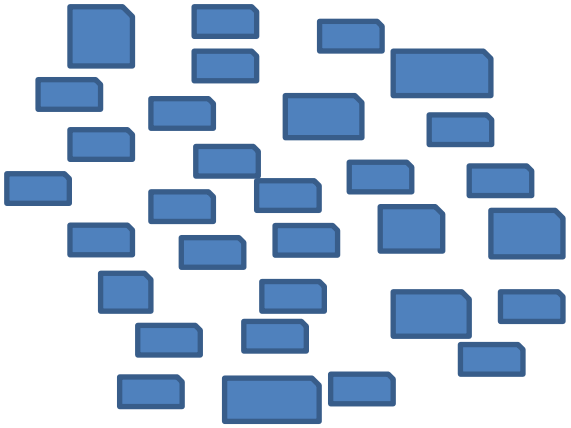
- Threads may disturb each other as they need a shared cache, memory bus, network bandwidth, disk,….

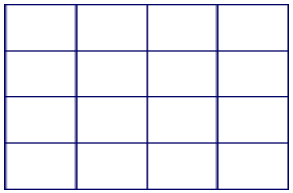
- **Collisions: Requesting the same resource**
  Example: several cores share the same FPU.
  - The requests are queued.
  - The waiting cores waste energy.

- **Destructive Interference: Causing more work to each other**
  Example: threads pollute each others' cache
  - Causes each thread to waste more energy to carry out the same workload

Idea: Resource-Aware O.S. Scheduler

- Motivation
  - Contention on resources (e.g. cache) wastes energy and usually degrades performance
- Proposal
  - Dynamically tune the workload, in order to ***minimize the contention*** on shared resources, by balancing the system.



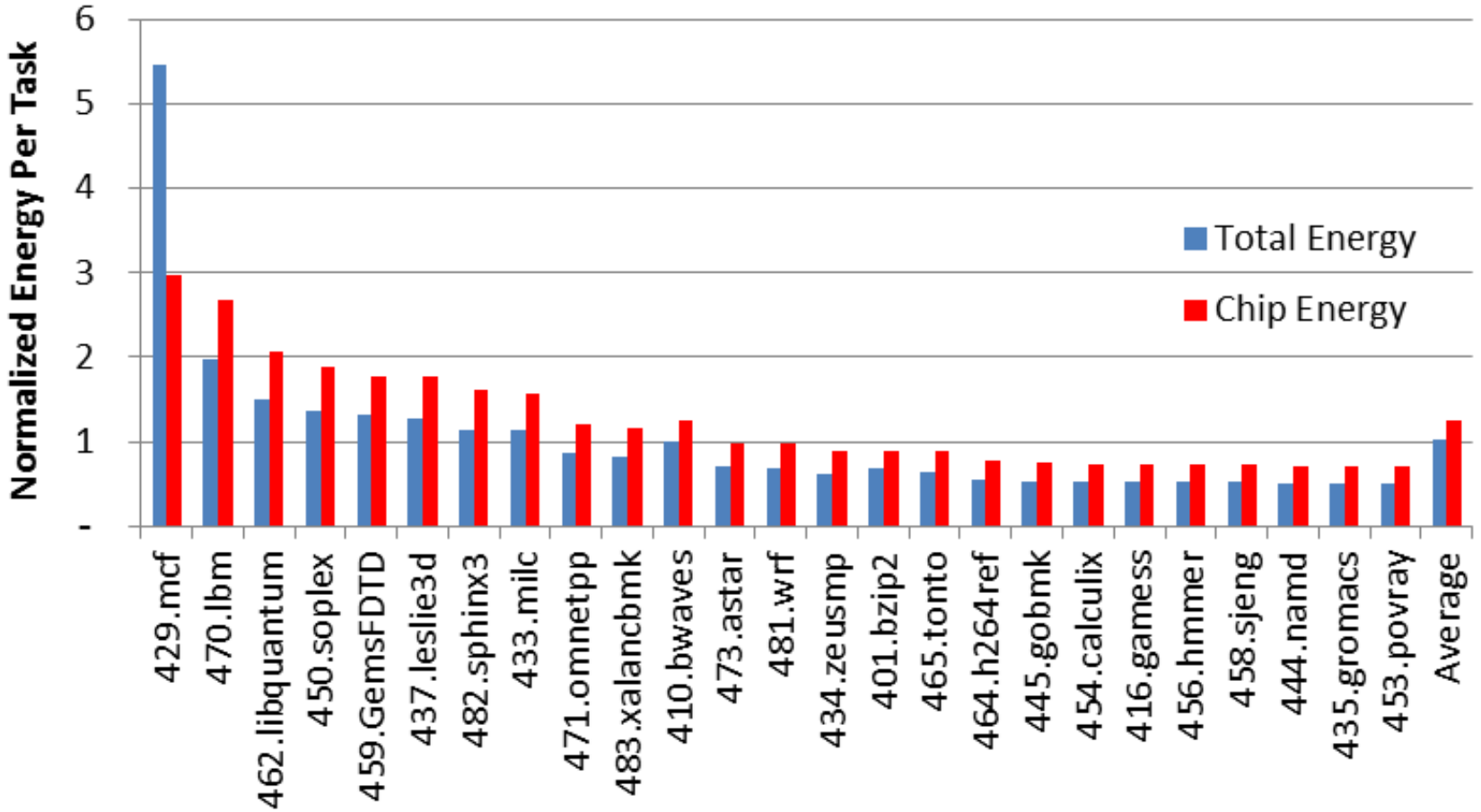


36

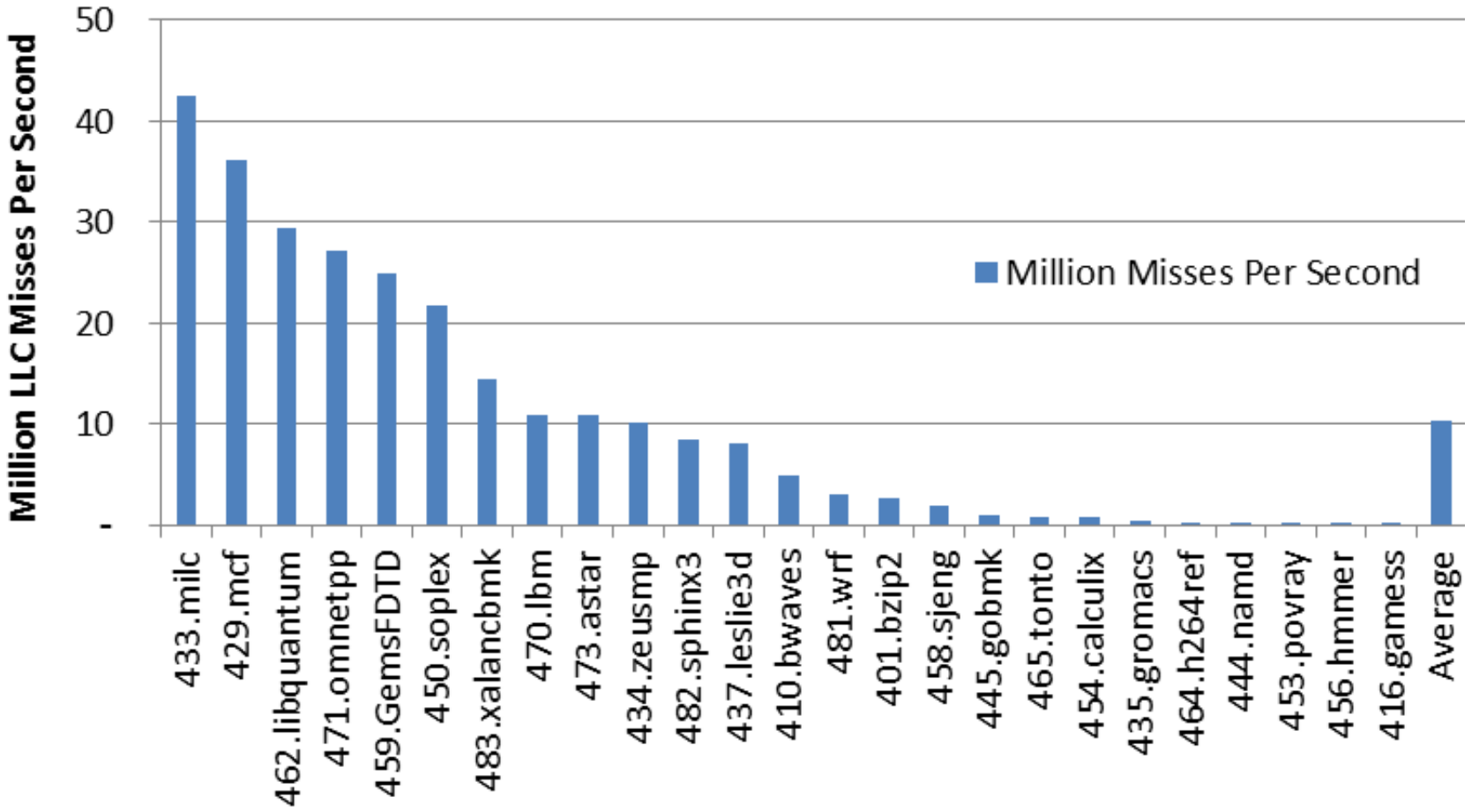
Experiments with a BW-hungry program

- Single task
  - 5.58 sec, 132 Joules
- When run x4 times sequentially
  - 22.3 sec, 526 Joules
- When run x4 times in parallel (4 core i5-2500)
  - 27.86 sec (**+25%**), 1368 Joules (**+160%**) – over sequential

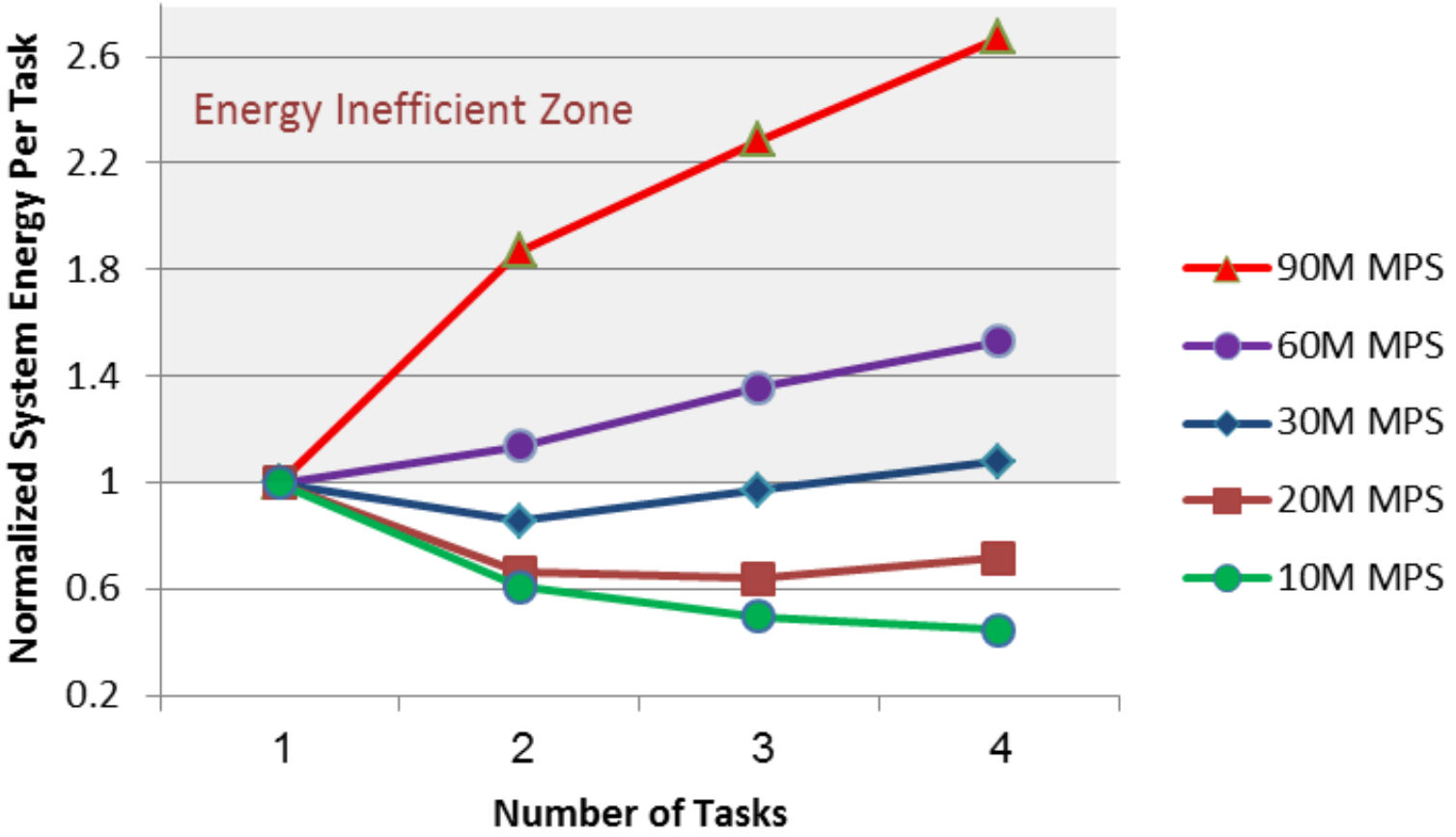
Normalized Energy per Task of Four Identical SPEC-CPU2006 benchmarks (baseline)



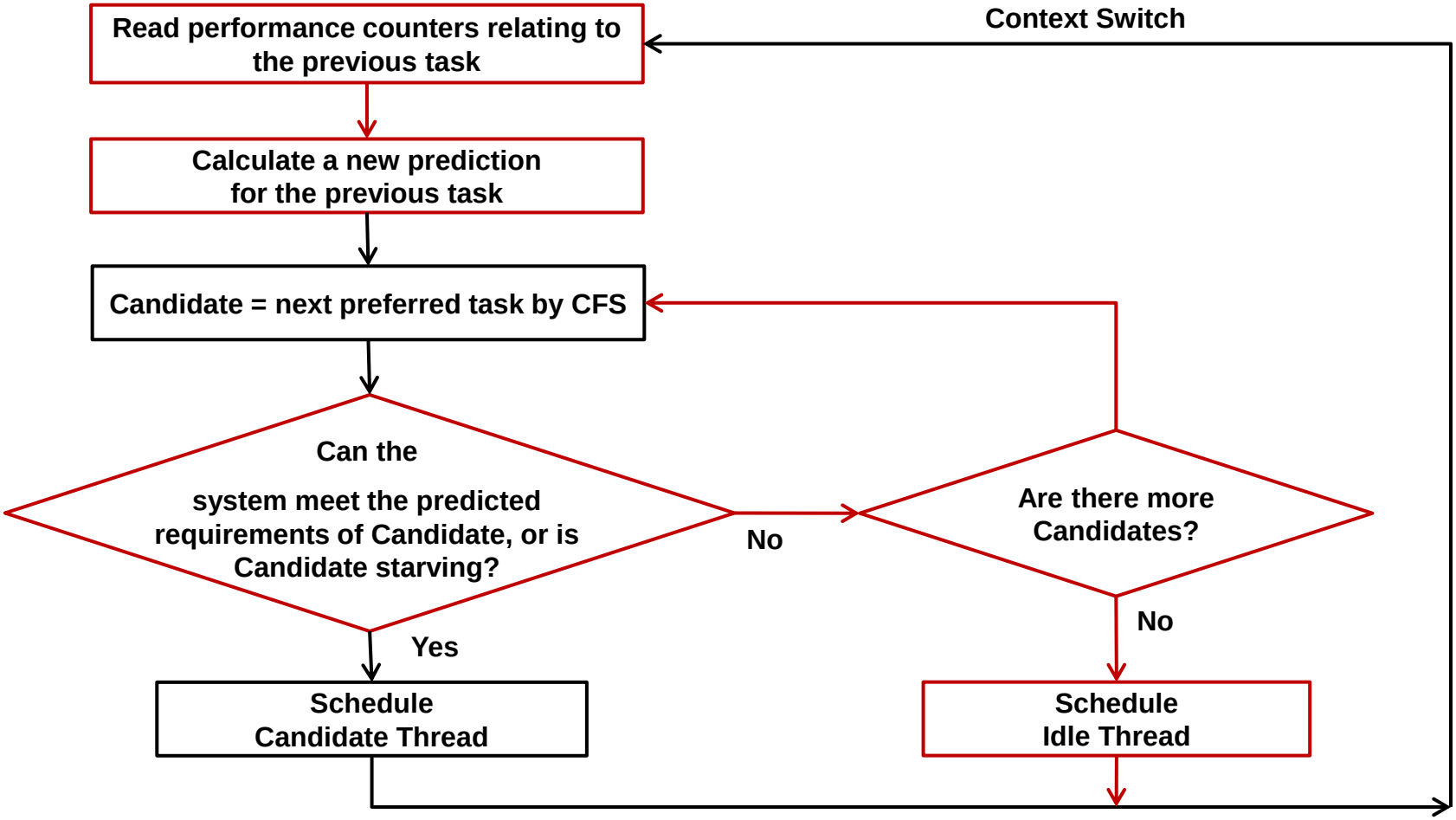
Looking into *Memory Bus Usage*



System Energy Per Task
for a bandwidth-hungry synthetic benchmark



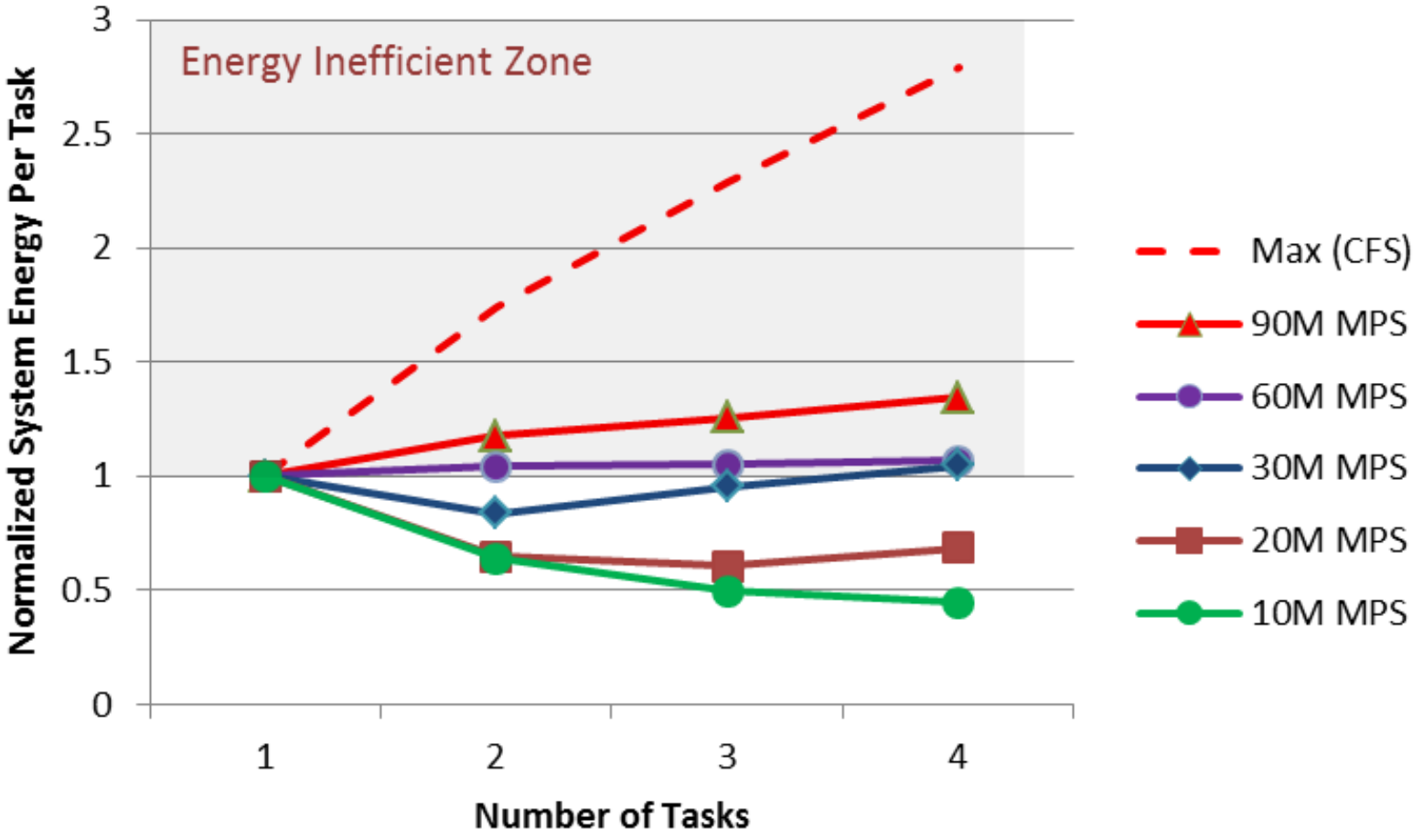
Proposed Scheduler



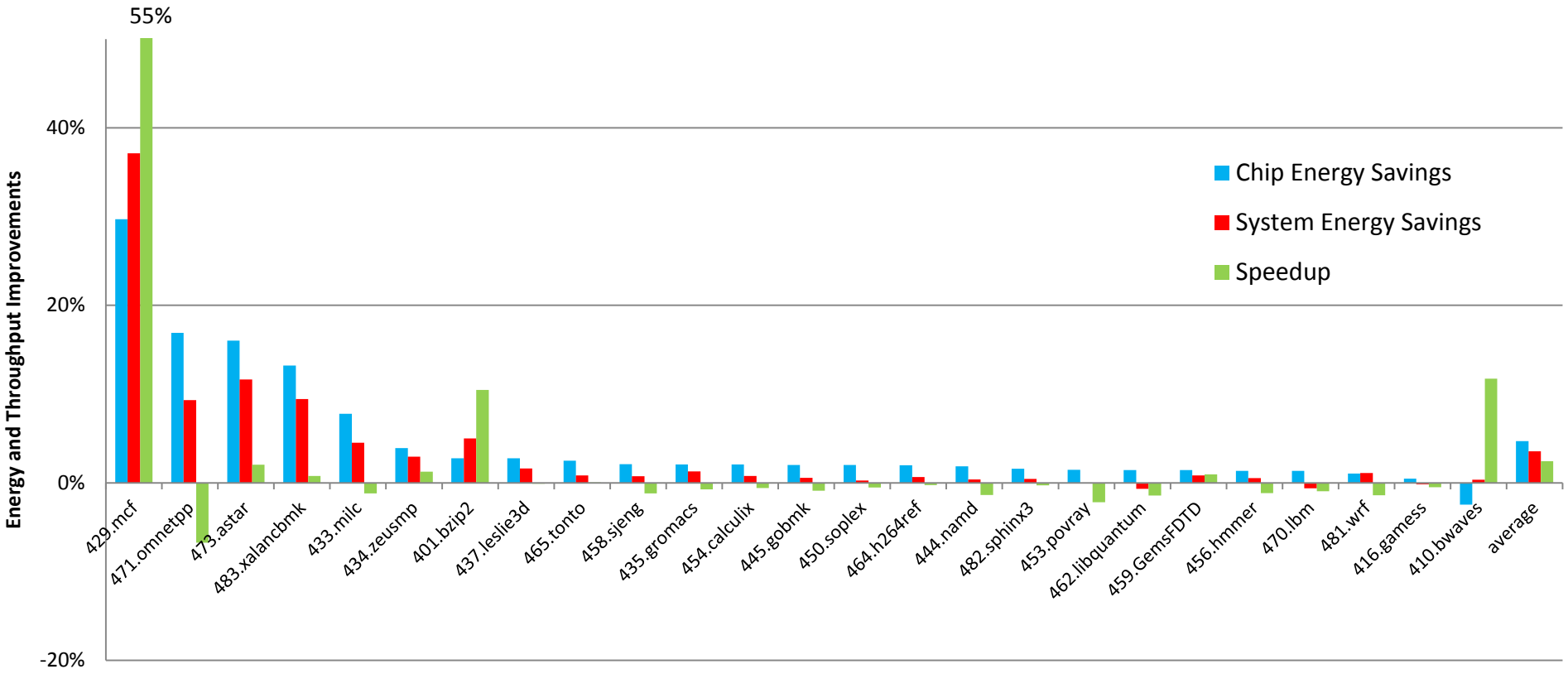
The New Scheduler with the BW-hungry program

- Single task
  - 5.58 sec, 132 Joules
- When run x4 times sequentially
  - 22.3 sec, 526 Joules
- When run x4 times in parallel (4 core i5-2500)
  - 27.86 sec (**+25%**), 1368 Joules (**+160%**) – over sequential
- X4 Using the new scheduler with memory bandwidth limitation enforcement
  - 23.71 sec (**+6%**), 569 Joules (**+8%**) – over sequential performance
- X4 Resource Aware Scheduler vs. Baseline scheduler
  - **17.5% speedup, 58% energy reduction**

System Energy Per Task (our scheduler)



Improvements in SPEC-CPU2006 Benchmarks



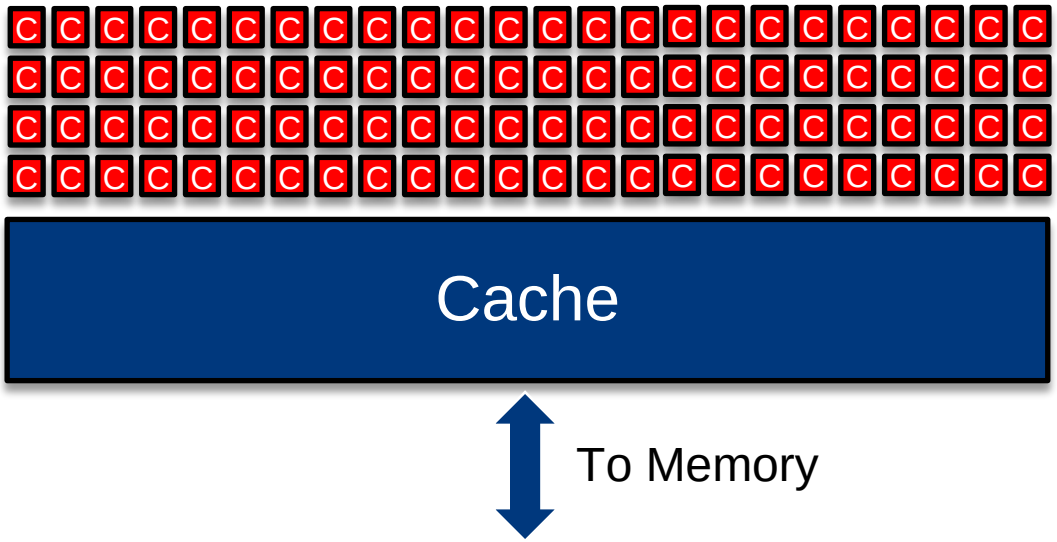
So:

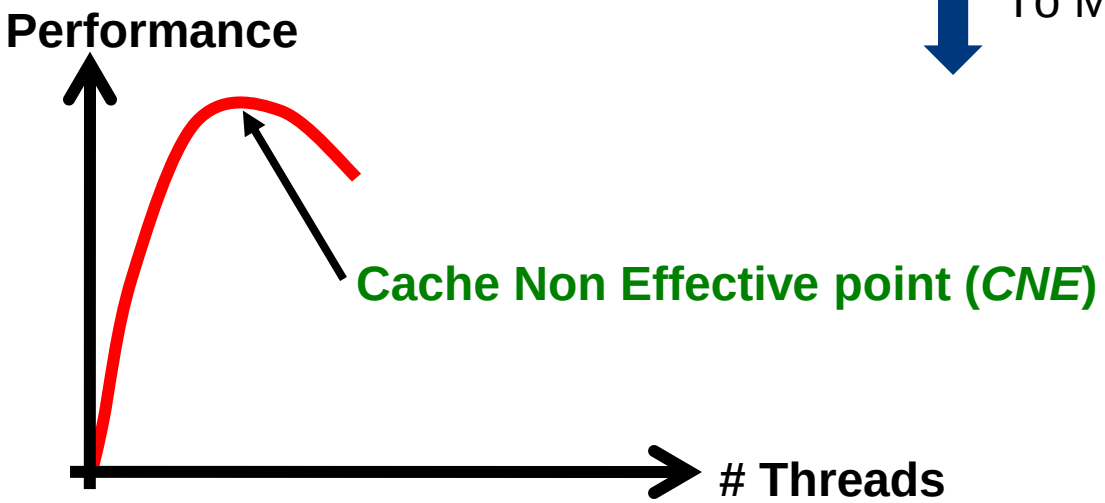
Waste of power at the system level can be reduced by system management

Back to System Architecture

Cache Machines

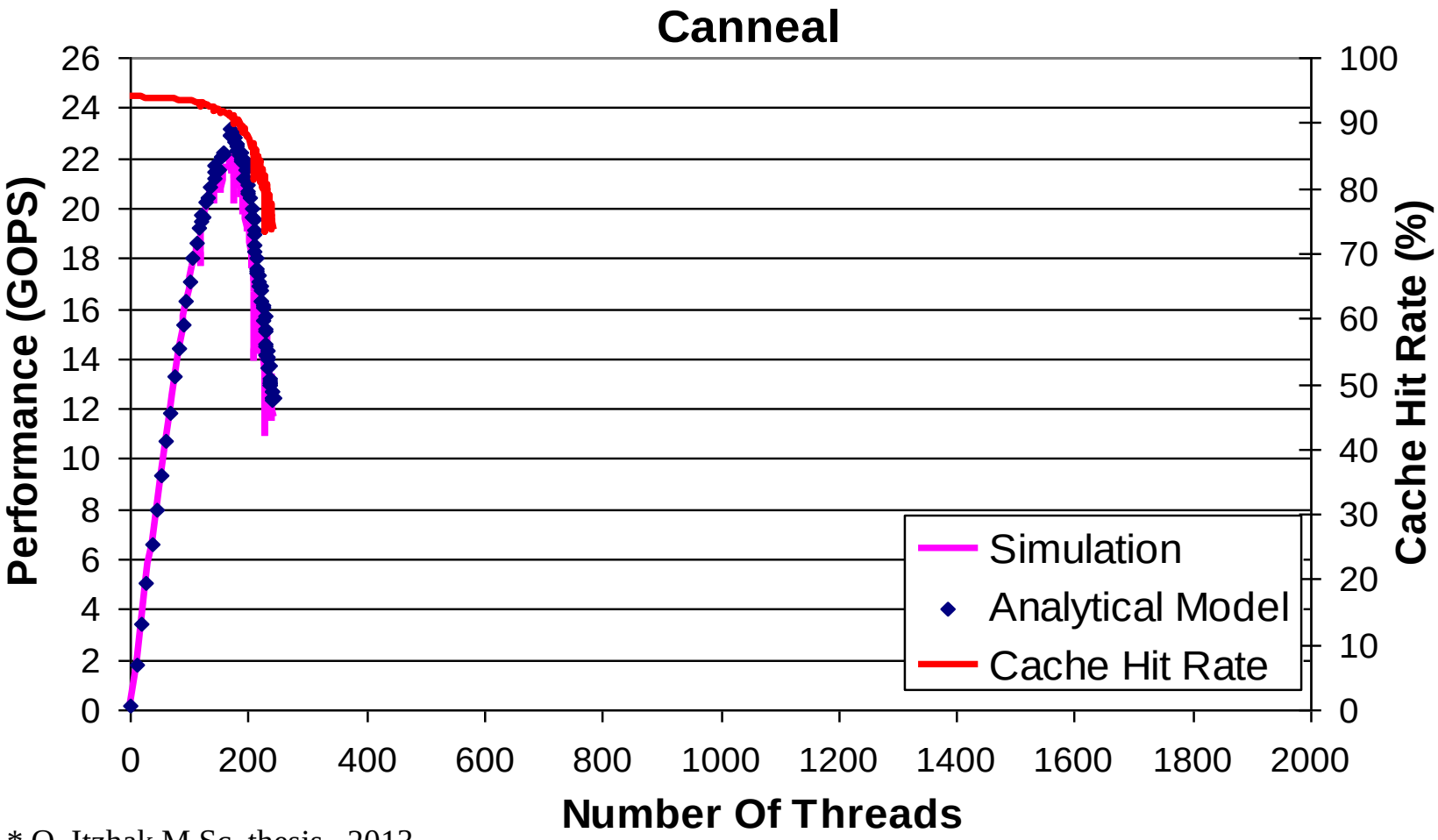
- Many cores (each may have its private L1) behind a shared cache





Poor Parallelism in Some MT Benchmarks

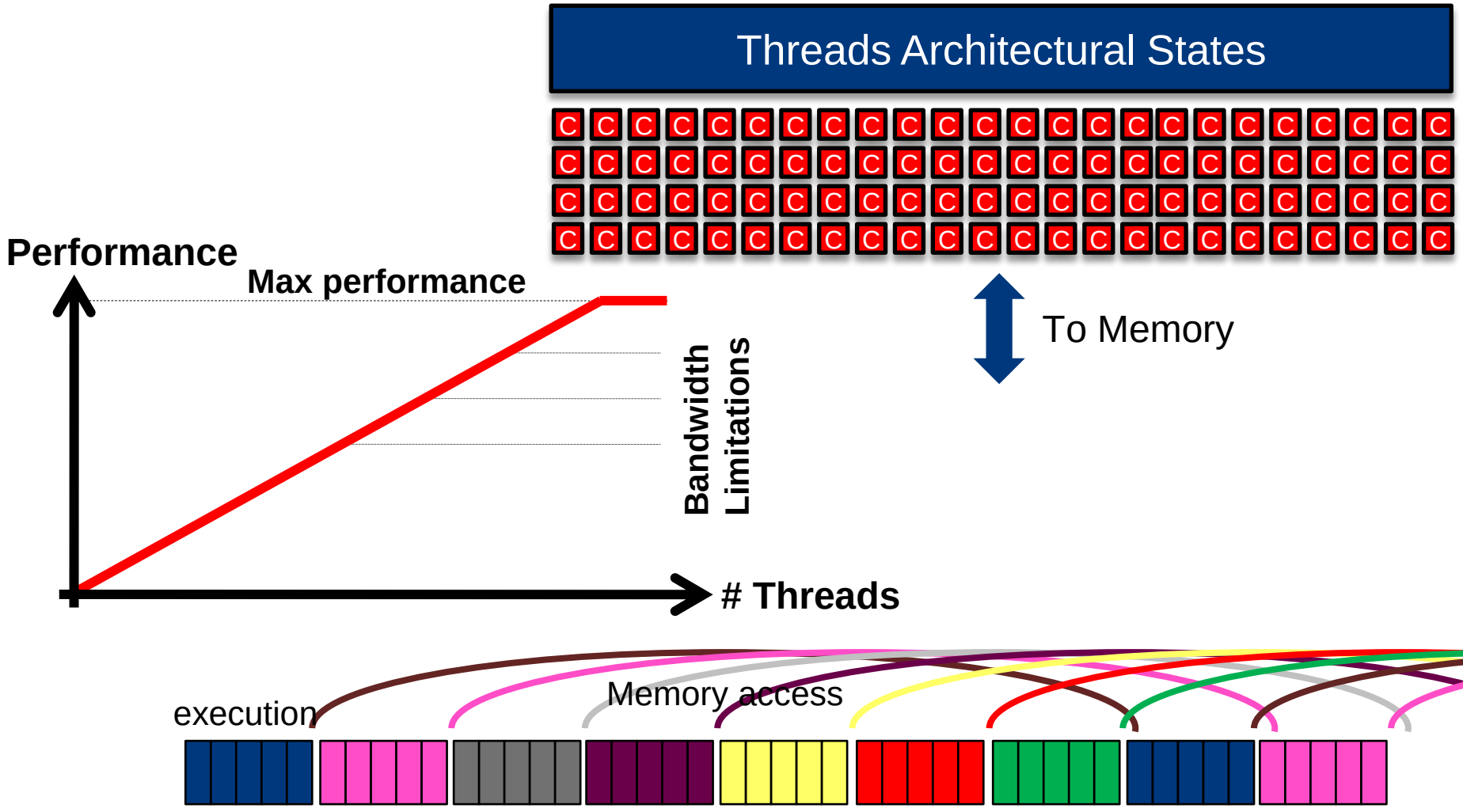
- Simulation results from the PARSEC workloads kit



* O. Itzhak M.Sc. thesis., 2013.

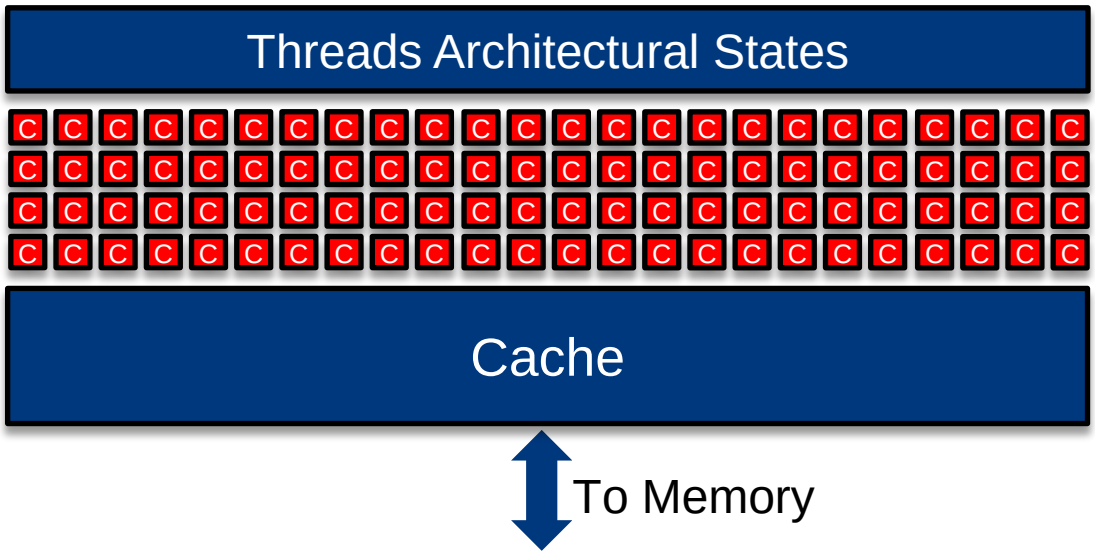
Multi-Thread Machines (e.g. GPGPU)

- Memory latency shielded by multiple thread execution



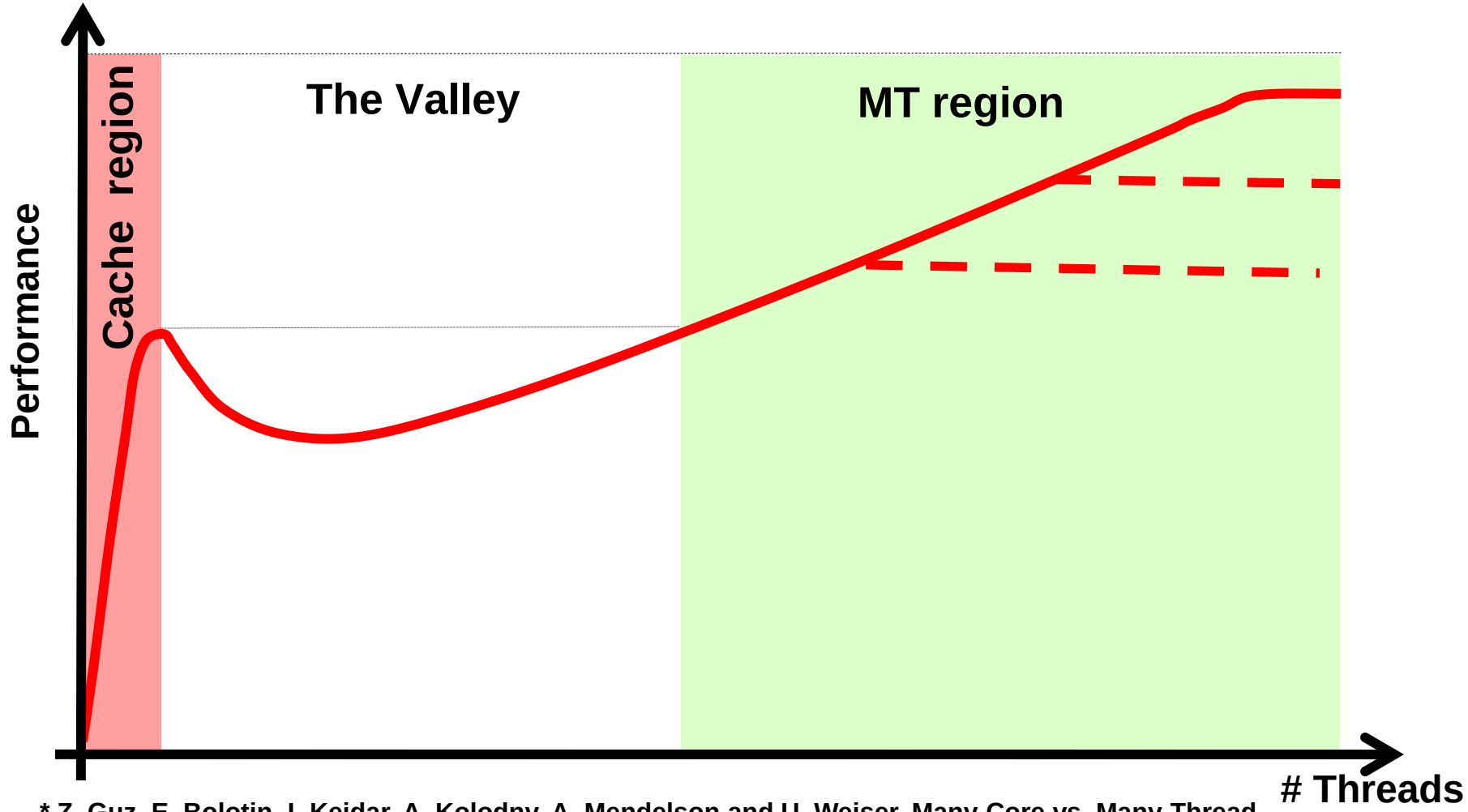
A Unified Machine Model

- Use both cache and many threads to shield memory access
  - The uniform framework renders the comparison meaningful
  - We derive simple, parameterized equations for performance, power, BW,..



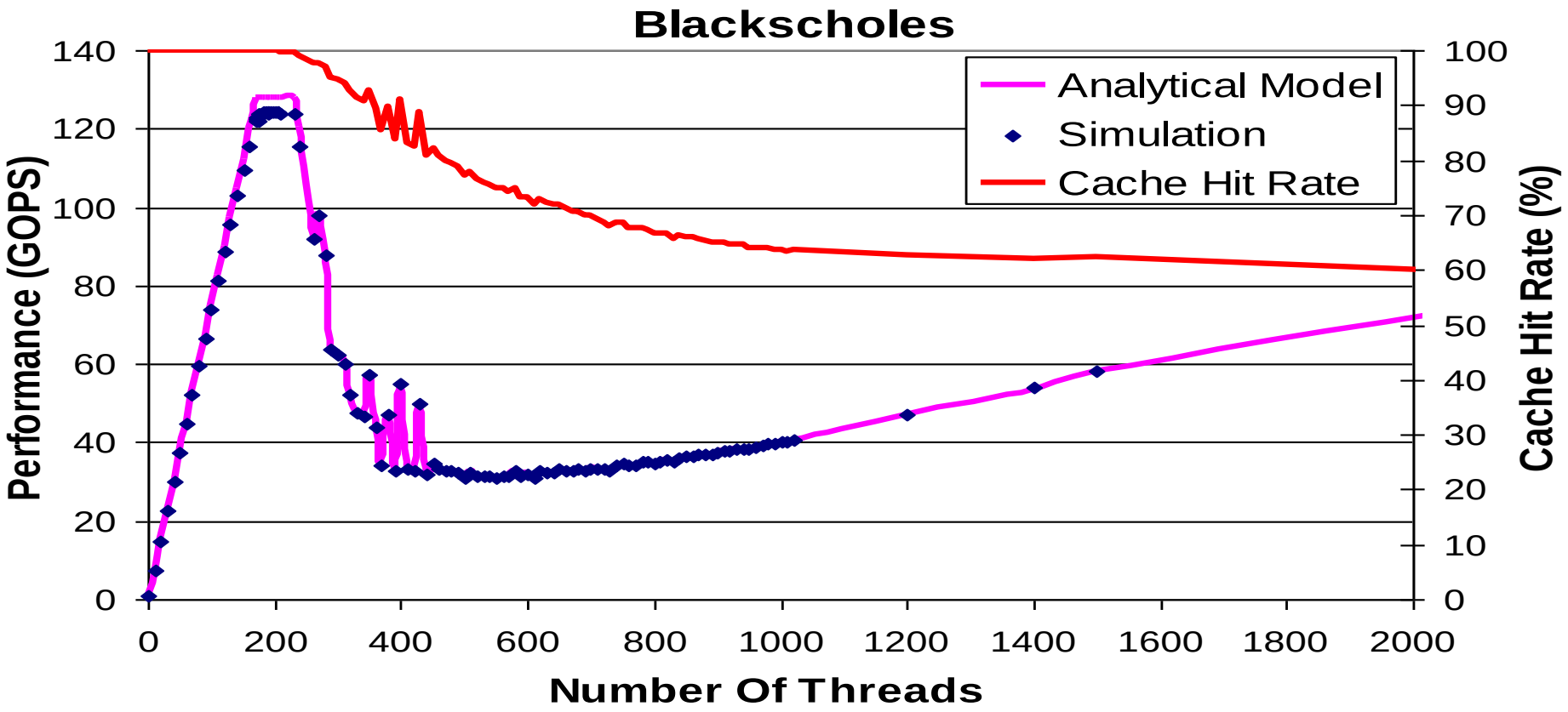
Unified Machine Performance

- 3 regions: Cache efficiency region, The Valley, MT efficiency region

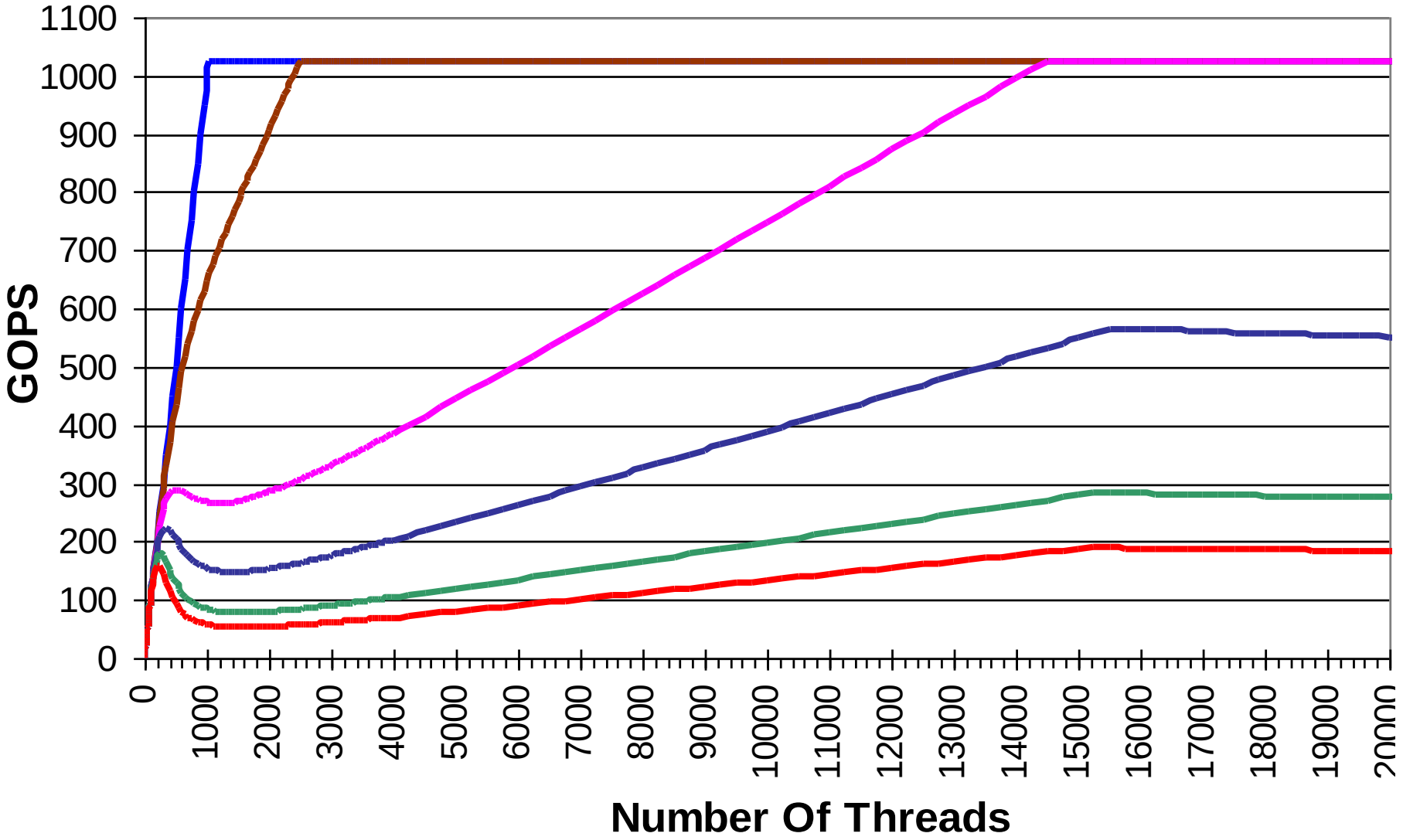


* Z. Guz, E. Bolotin, I. Keidar, A. Kolodny, A. Mendelson and U. Weiser, Many-Core vs. Many-Thread Machines: Stay Away From the Valley", IEEE Computer Architecture Letters, Volume 8, Issue 1, Jan. 2009

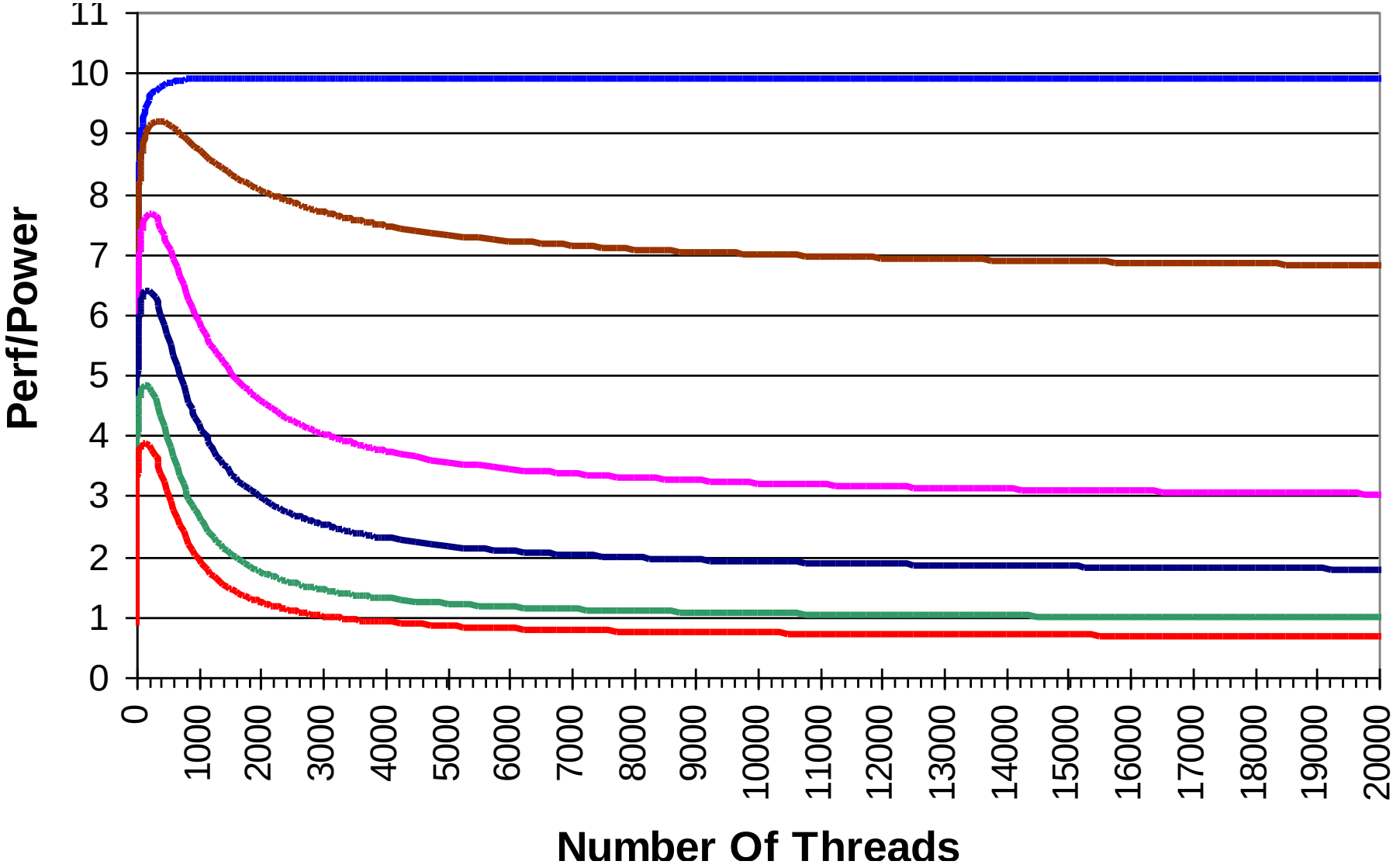
Validation of the model 1



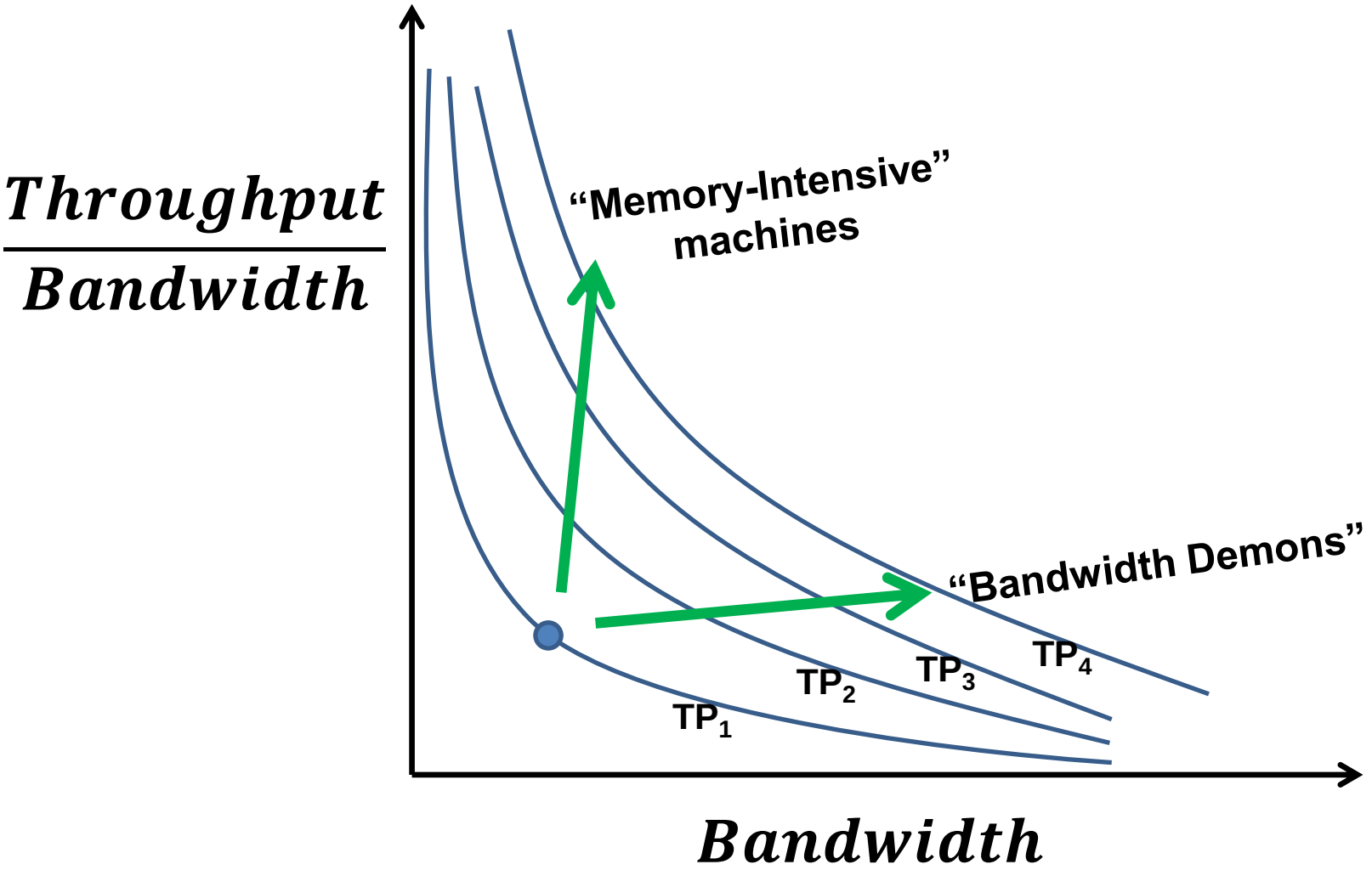
Performance depends on the fraction of memory instructions



Performance/Power 1/(Energy Per Instruction)



Let's Look at Equi-Throughput Curves



- Reducing BW (i.e. power) can be achieved by climbing up a constant-throughput-curve
- ➔ increase on-die-memory (e.g. innovative cache, new ideas….?)

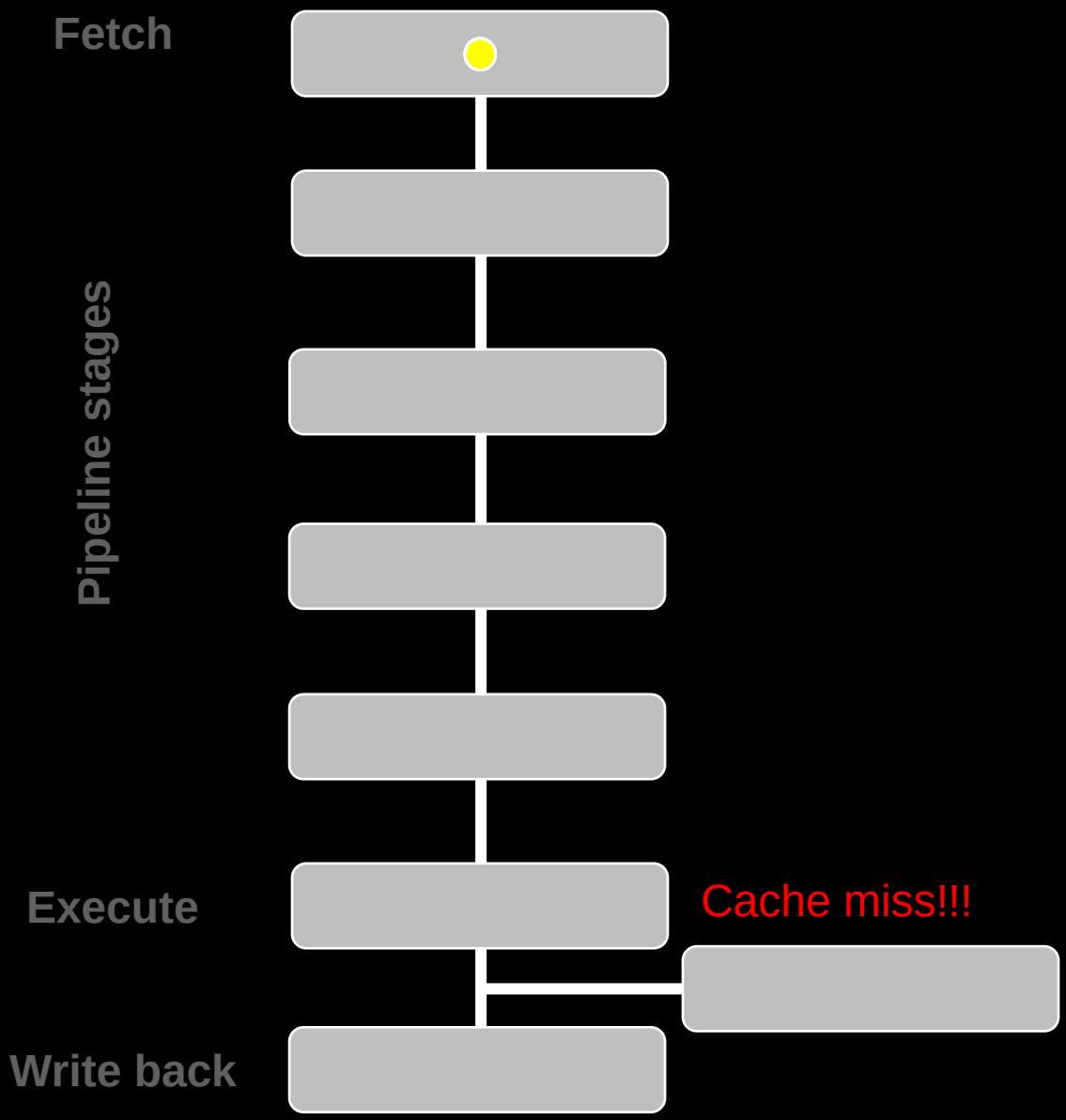
Proposal: Memory-Intensive Architectures

- Embed memory in execution units
- Add computational capabilities in memory

Example of a Memory-Intensive Architecture: Continuous Flow Multithreading (CFMT)

Switch on Event Multithreading

Example- processor pipeline

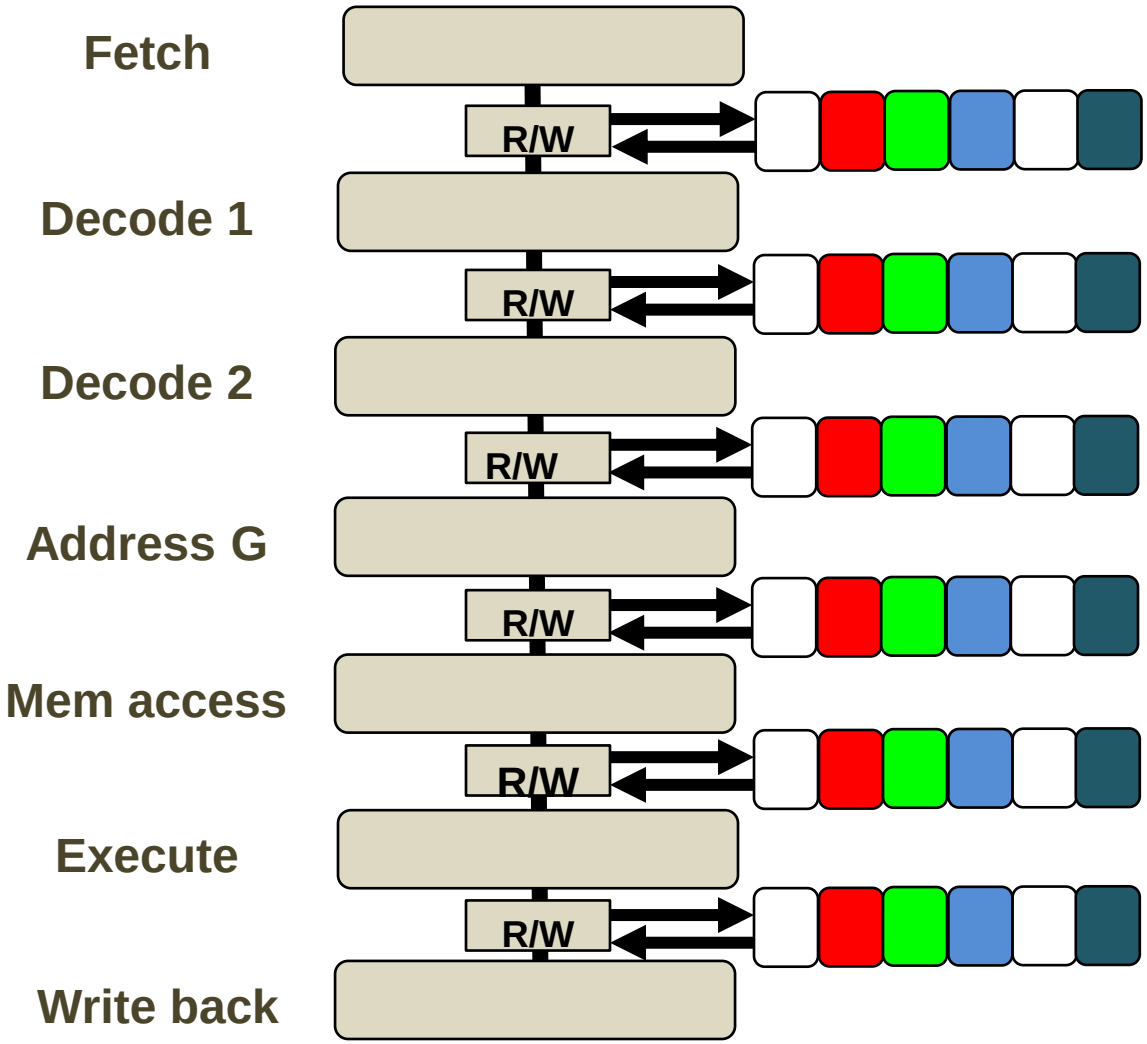


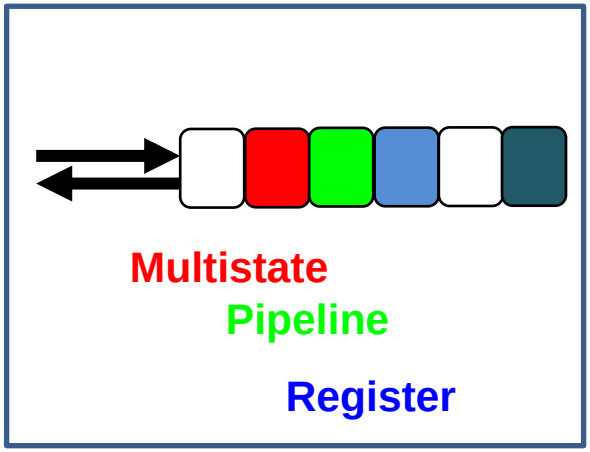
Thread A

Thread B

Thread C

Using Multistate Pipeline Registers

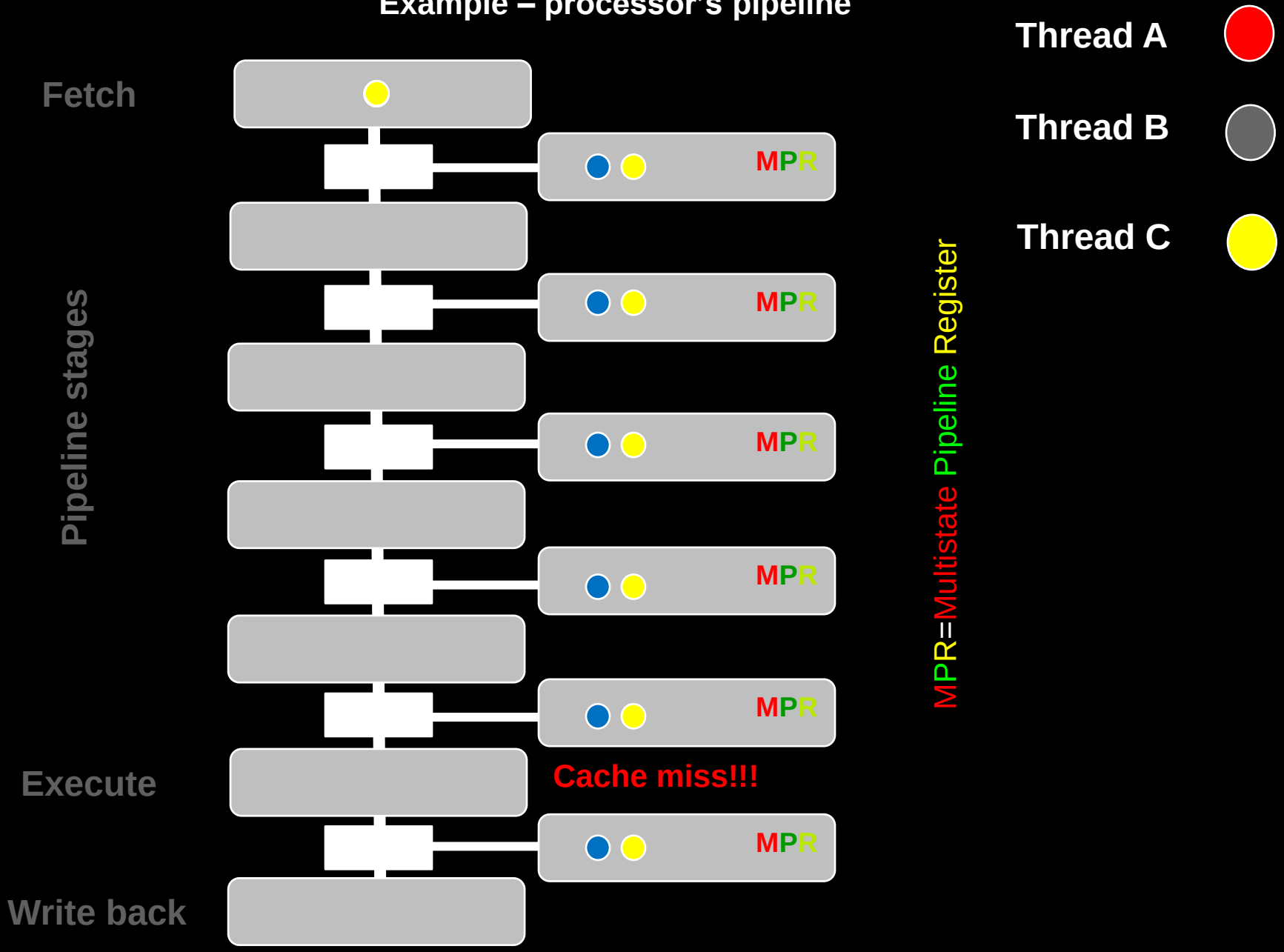




S. Kvatinsky et al., "Memristor-based Multithreading," CAL 2013

Continues Flow MT (CFMT)

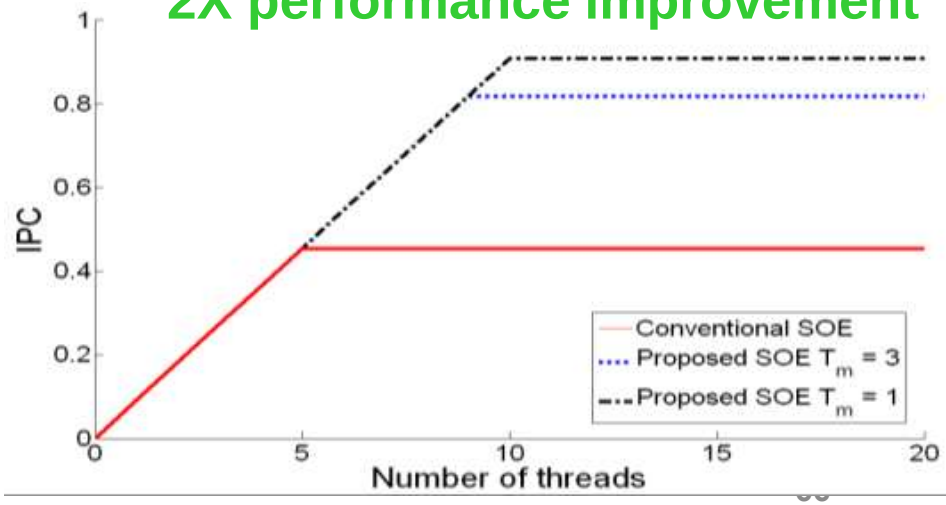
Example – processor's pipeline



CFMT – A Novel µArchitecture

- The simplicity of SoE MT
- Using novel memory structure - MPR
- No pipeline flush:
  - Enhance performance
  - Reduce power

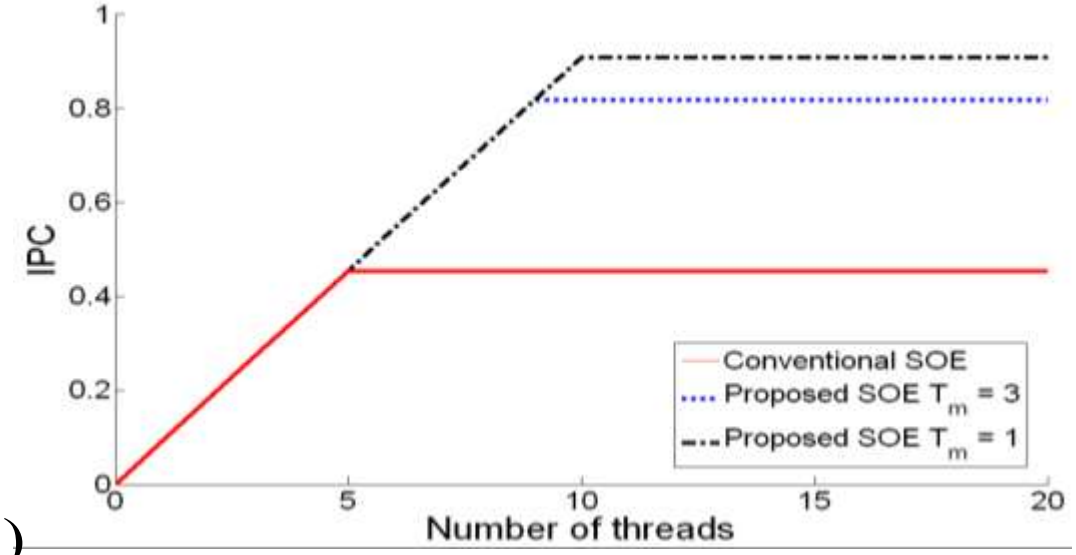
2X performance improvement



S. Kvatinsky et al., "Memristor-based Multithreading," CAL 2013

CFMT Performance Analysis

- 2X theoretical performance speedup
- Simulations show similar improvements

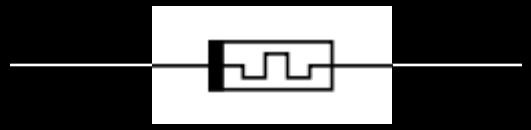


$$CPI = \begin{cases} \dfrac{CPI_{ideal} + P_m \cdot r_m \cdot MR(n)}{n}, unsaturated \\ CPI_{ideal} + P_s \cdot r_m \cdot MR(n), saturation \end{cases}$$

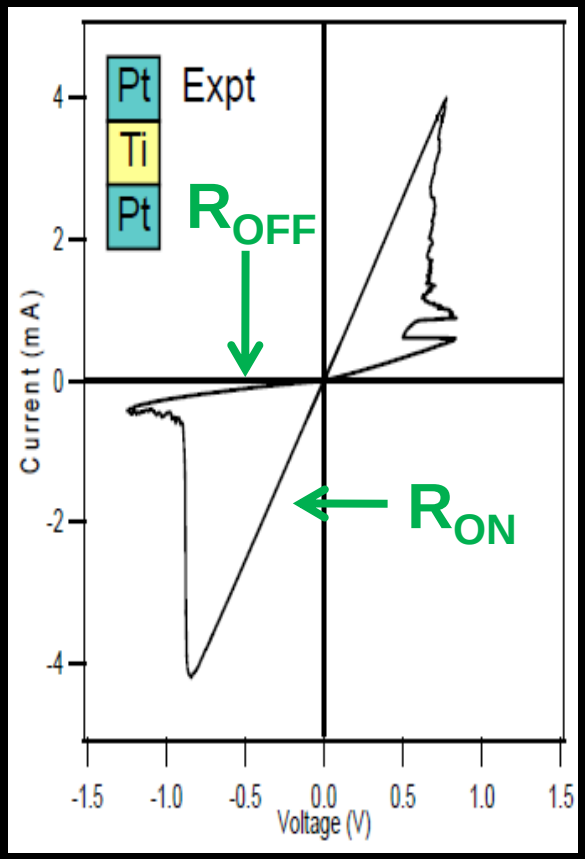
Memristor Technology:
Enabler for Memory-Intensive Architecture

What is a Memristor?

- 2-terminal resistive nonvolatile device



- Device's resistivity depends on past electrical current (based on resistance switching effects)
- Device is constructed of 2 metal layers with oxide in between (e.g. T_iO_2)

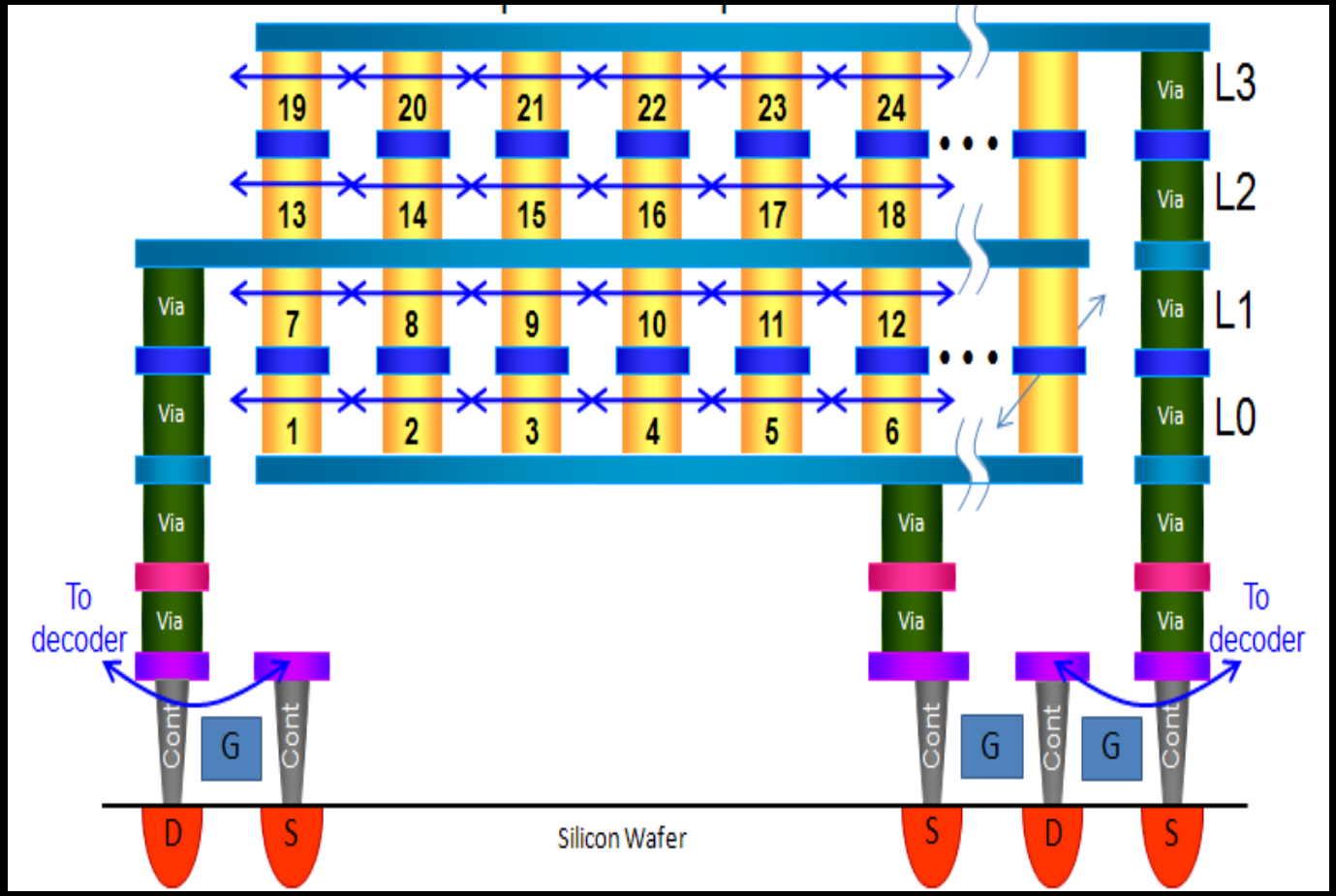


Current [mA]

Voltage [V]

Memristor Microarchitecture "Vision"

- Layers of memory cells above logic



Advantages of Memristors

- Large amount of memory cells
- Very close to logic
- Non volatile
  - No need for power to “stay alive”
- Small size
- Fast
- No leakage

Sea of Memory Cells - Ideas

- Conventional vs. Out of the box

- New types of caches
- Increase on-die prediction structures
- Continues Flow Multithreading (improved SoE MT)
- Instruction queues
- Instruction reuse (memorization)
- Enhance Multithreading architecture (Graphics like)
- Computation at the memory level

Summary

- System Level and Physical issues are converging
  - On-chip distances dominate everything
- Lots of opportunities to save power at system level
- In a given architecture:
  system management challenge
- In future systems:
  system architecture challenge

Thanks

This talk is based on joint work with many students and collaborators:

Nir Magen
Oved Itzhak
Shahar Kvatinsky
Tomer Morad
Yaniv Ben-Itzhak
Yoni Aizik
Yuval Nacson
Zigi Walter
Zvika Guz

Avi Mendelson
Idit Keidar
Uri Weiser
Yoav Etzion