18-600 Foundations of Computer Systems

Lecture 22: "Performance and Power Iron Laws"

John P. Shen & Zhiyi Yu
November 16, 2016

The "Science" of Computer Architecture

- Recommended References:
  - "Energy per Instruction Trends in Intel® Microprocessors," by Ed Grochowski, Murali Annavaram, 2006.
  - "Mitigating Amdahl's Law through EPI Throttling," by M. Annavaram, E. Grochowski, J. Shen. In 32nd ISCA 2005.

Electrical & Computer ENGINEERING
Carnegie Mellon University

18-600 Foundations of Computer Systems

Lecture 22:
"Performance and Power Iron Laws"

A. Law #1 – Processor (Latency) Performance
B. Law #2 – Multiprocessor (Throughput) Performance
C. Law #3 – Multiprocessor Performance Scalability
D. Law #4 – Algorithm and Performance
E. Law #5 – Power and Performance
F. Power Reduction & Energy Efficiency

Law #1 – Processor (Latency) Performance

- Time to execute a program: T (latency)

$$T = \frac{instructions}{program} \times \frac{cycles}{instruction} \times \frac{time}{cycle}$$

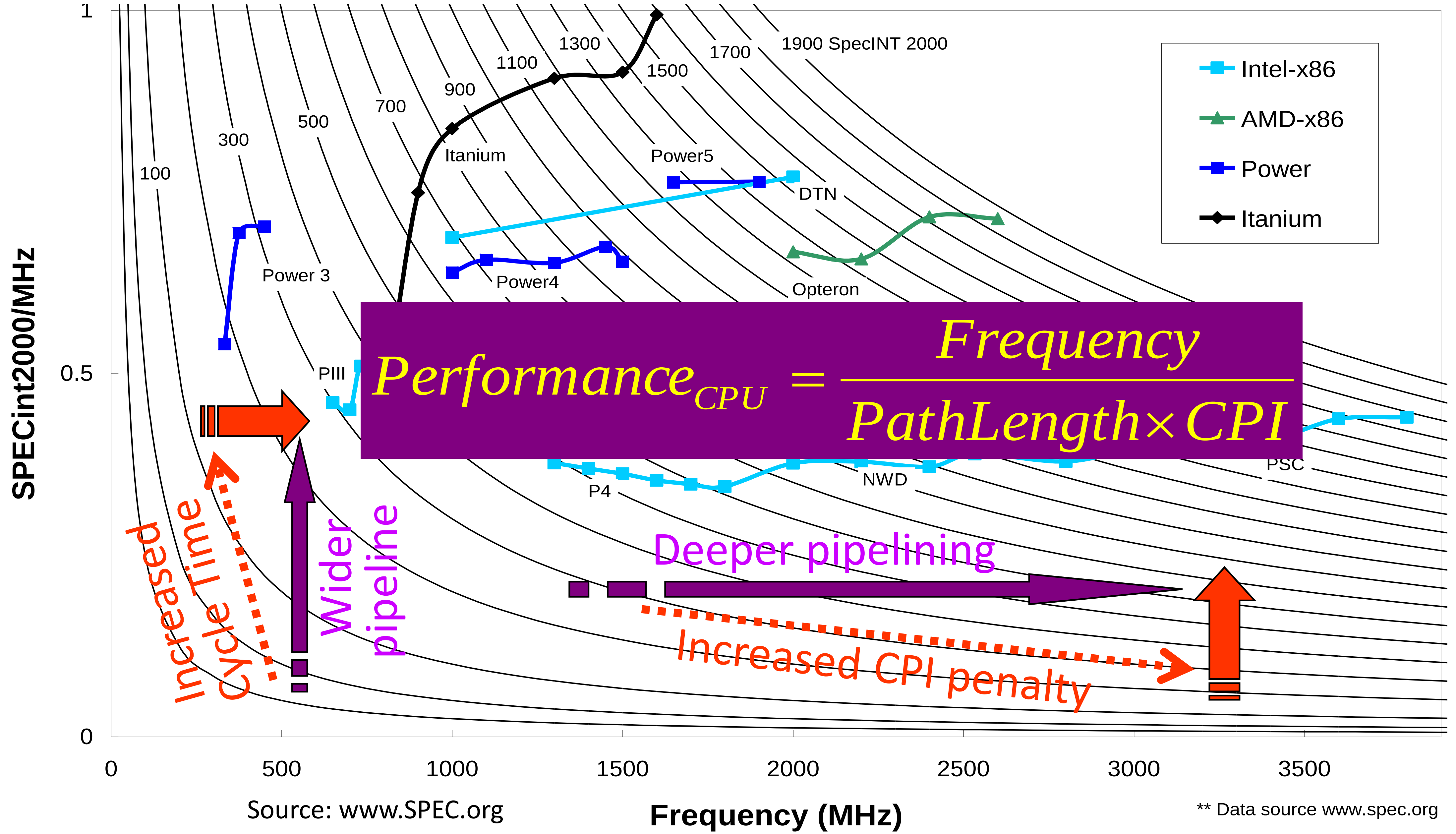
$$T = PathLength \times CPI \times CycleTime$$

- Processor performance: Perf = 1/T

$$Perf_{CPU} = \frac{1}{PathLength \times CPI \times CycleTime} = \boxed{\frac{Frequency}{PathLength \times CPI}}$$

[John DeVale & Bryan Black, 2005]

SPECint (Latency) Performance of Processors



$$Performance_{CPU} = \frac{Frequency}{PathLength \times CPI}$$

Latency vs. Throughput Performance

- **Reduce Latency of Application**
  - Uni-processor, Single Program
  - Target Single-Thread (ST) Performance
  - Examples: SPEC, PC and Workstations

- **Increase Throughput of System**
  - Multi-processors/Multi-cores, Many Threads/Tasks
  - Target Multi-threaded/Multi-tasking Throughput
  - Example: Database Transaction Processing

Ideal Multiprocessor (Throughput) Performance

- Time to process a thread/task: T (latency)

$$T = \frac{instructions}{thread} \times \frac{cycles}{instruction} \times \frac{time}{cycle}$$

$$T = PathLength \times CPI \times CycleTime$$

- MP (n processors) performance: Perf = 1/T

$$Perf_{MP} = \frac{n \times 1}{PathLength \times CPI \times CycleTime} = \boxed{\frac{n \times Frequency}{PathLength \times CPI}}$$

Law #2 – Multiprocessor/Multicore Performance

- Multi-Core Performance:

$$Perf_{MC} = \boxed{\frac{n \times Frequency}{PL(n) \times CPI(n)}}$$

- Can Improve $Perf_{MC}$ by:
  - Increasing: n (no. of CPUs or cores)
  - Increasing: $Frequency$ (CPU clock frequency)
  - Decreasing: PL (dynamic instruction count)
  - Decreasing: CPI (cycles/instruction)

A Multiprocessor Performance Scalability Model

- Multi-Core Speedup:

$$Speedup(n)_{MC} = \frac{\dfrac{n \times Frequency}{PL(n) \times CPI(n)}}{\dfrac{Frequency}{PL(1) \times CPI(1)}} = \boxed{n \times \left(\frac{PL_1 \times CPI_1}{PL(n) \times CPI(n)} \right)}$$

- A Rigorous Scalability Model (proposal):

$$PL(n) \times CPI(n) \cong \boxed{n^x PL_1 \times n^y CPI_1}$$

Law #3 – Multicore Performance Scalability

- **Multi-Core Speedup:**

$$Speedup(n)_{MC} = n \times \left(\frac{PL_1 \times CPI_1}{n^x PL_1 \times n^y CPI_1} \right) = \boxed{\left(\frac{n}{n^x \times n^y} \right)}$$

- **Scalability <u>Impedance</u> Functions:**

$$\begin{cases} n^x \equiv PL(n) \text{ Impedance Function} \\ n^y \equiv CPI(n) \text{ Impedance Function} \end{cases}$$

$$\left(n^x \times n^y\right) = n^{(x+y)} \;\middle|\; \begin{cases} (x+y) = 1.0 \Rightarrow \text{No Speedup} \\ (x+y) = 0.0 \Rightarrow \text{No Impedance} \end{cases}$$

[Rich Hankins & Murali Annavaram, 2004]

Real World Example: Database Performance (OLTP)

- MP Database Performance: TPS

$$TPS = \frac{Transactions}{Second} = \boxed{\frac{n \times Frequency}{IPX(n) \times CPI(n)}}$$ [Law #2]

- Can Improve TPS by:
  - Increasing: n (no. of CPUs)
  - Increasing: $Frequency$ (CPU clock frequency)
  - Decreasing: IPX (instructions/transaction) == PL
  - Decreasing: CPI (cycles/instruction)

[Rich Hankins & Murali Annavaram, 2004]

4-way Multiprocessor Experimental Setup

Component	Intel® Xeon™ System	Intel® Itanium® 2 System
Processors	4-way SMP, 1.6GHz	4-way SMP, 900MHz
Caches	256KB L2, 1MB L3	256KB L2, 3MB L3
Operating System	Red Hat® AS 2.1	Red Hat® AS 2.1
Disks	24 data + 2 log	32 data + 1 log
Main Memory	4GB	16GB
Database	Oracle® 9ir2	Oracle® 10g
OS Large Page Size	4MB	256MB
SGA	3GB	14GB

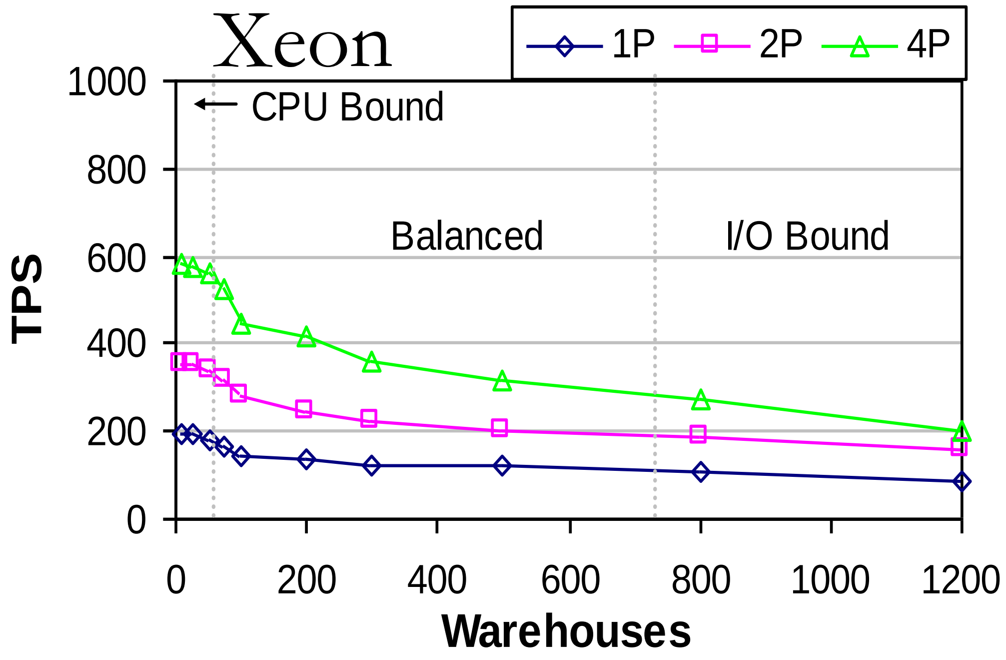
- Based on EMON Events
  - Separate User and OS components for each event
  - Use multiple long runs (20-min warm up, 10-min measurement)
  - Strive for standard deviation <5% for TPS & CPU utilization > 90%
  - Overall user execution time ~70-90%

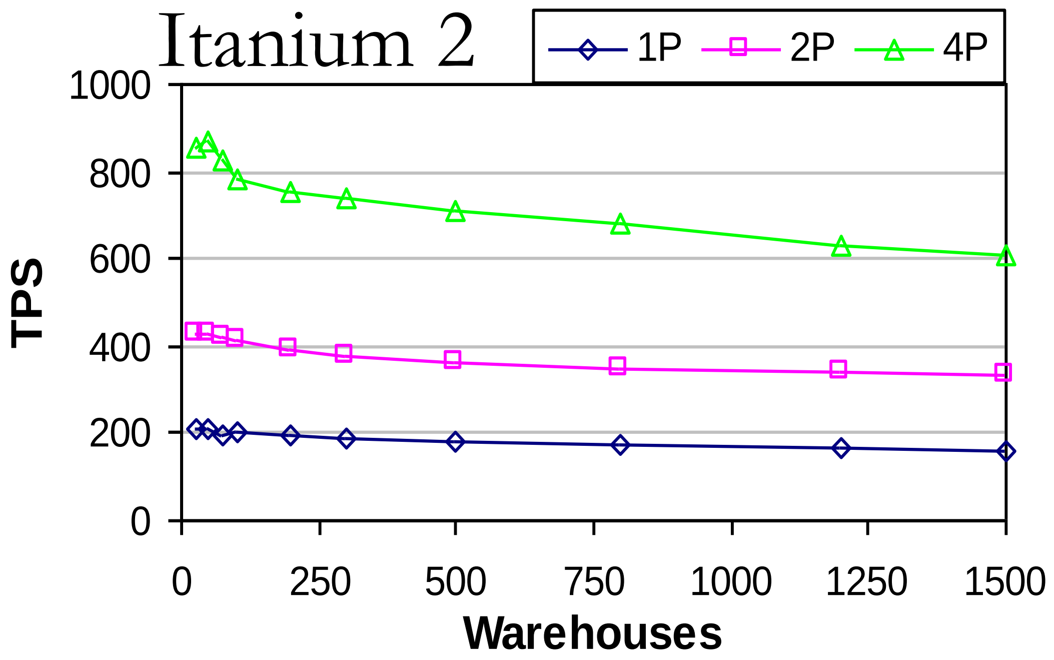
$$TPS = \frac{n \times Frequency}{IPX(n) \times CPI(n)}$$

[Rich Hankins & Murali Annavaram, 2004]

TPS: Throughput Scaling

$$TPS = \frac{n \times Frequency}{IPX(n) \times CPI(n)}$$





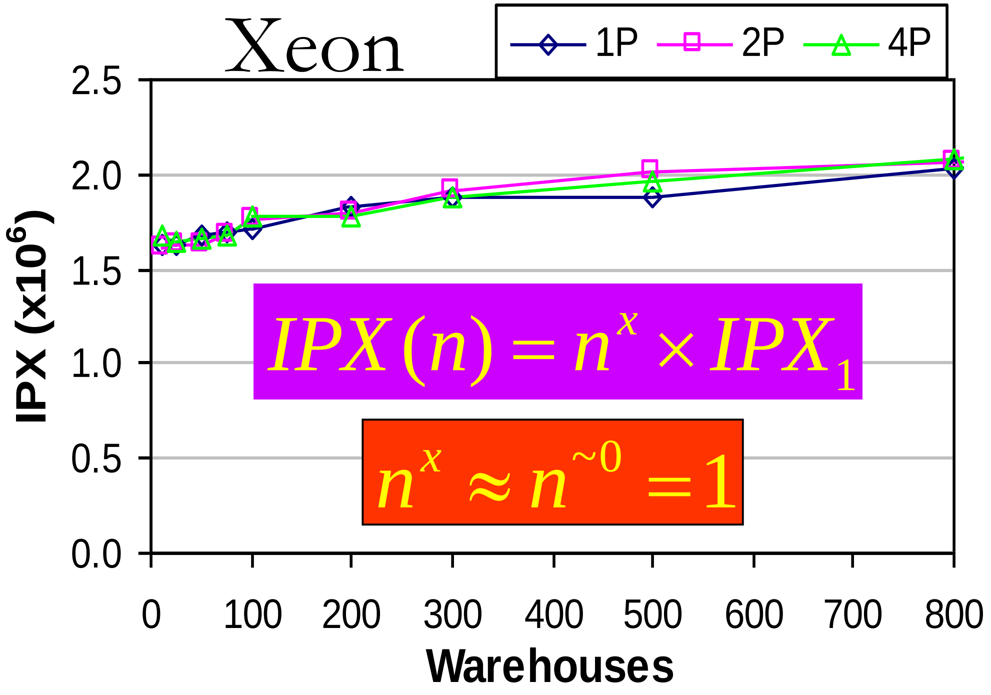
- TPS scales more linearly on Itanium2
  - Larger SGA implies slower I/O rate increase (Xeon I/O rate increases at 7KB/WH, & Itanium 2 at: 6KB/WH)
  - Bus utilization on Xeon higher than Itanium 2 (45% vs. 39%)
- Increasing I/O rate → more processes & context switches
  - Clients increase from 8 to 56 on Xeon & 4 to 64 on Itanium 2
  - OS time up to 20% on Xeon & only 10% on Itanium 2

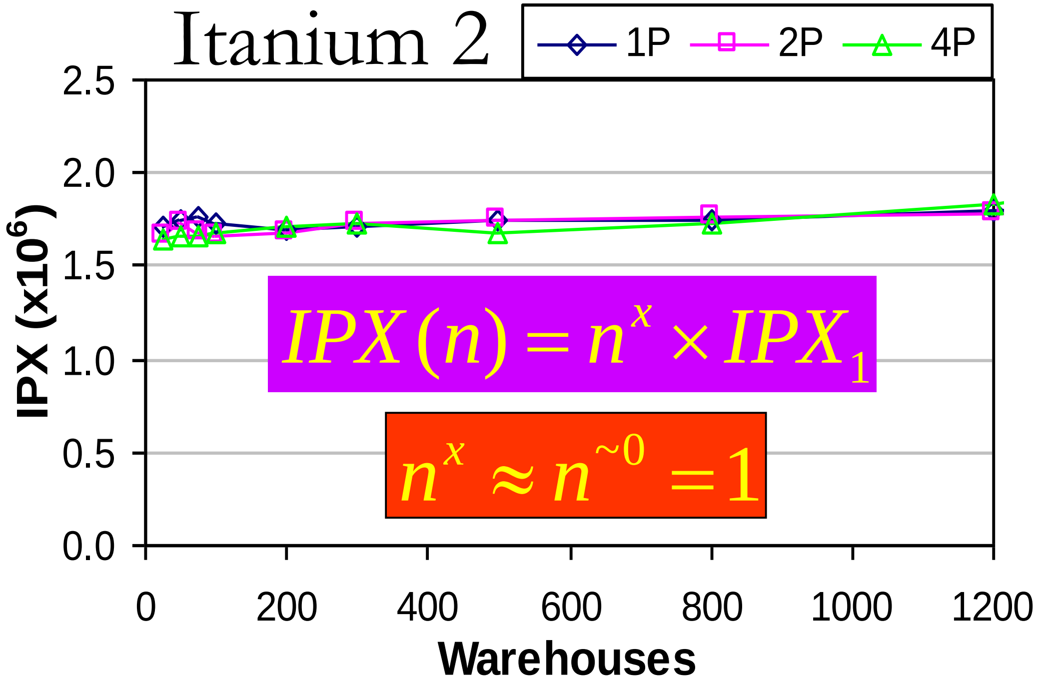
- Performance degrades with increased data set size (due to I/O rate)
- Performance improves with increased n (on IPF almost linear increase)

[Rich Hankins & Murali Annavaram, 2004]

IPX: Path-Length Scaling

$$TPS = \frac{n \times Frequency}{IPX(n) \times CPI(n)}$$





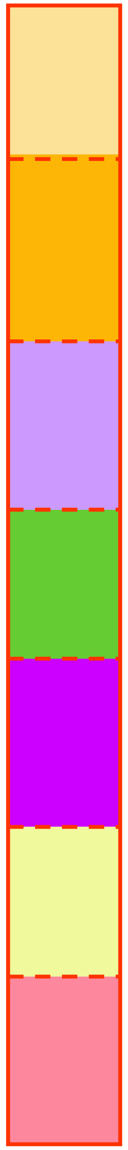
- Growth in IPX (quite linear) attributed to OS IPX increase
  - More I/O, more context switching
- User level IPX remains relatively constant in both systems
  - Code path through Oracle relatively constant
- Excluding NOPS, IPX at 25 WH similar for both systems!
- IPX growth less pronounced on Itanium 2

- IPX increases with increased data set size (Xeon)
- But IPX(n) doesn't increase much with increased n

Path-Length Contributions

$$TPS = \frac{n \times Frequency}{IPX(n) \times CPI(n)}$$

$$PL(n) \begin{cases} OS \begin{cases} I/O \\ Page\ Fault \\ Context\ Switch \end{cases} \\ APP \begin{cases} Original\ \ Code \\ Speculation \\ Replay \\ Synchronization \end{cases} \end{cases}$$

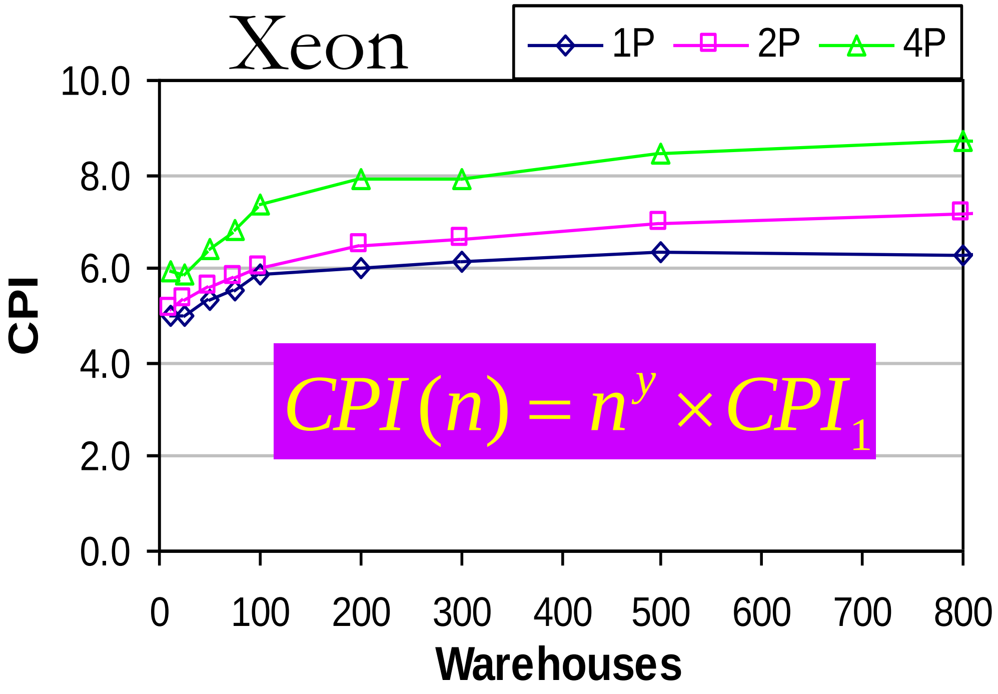


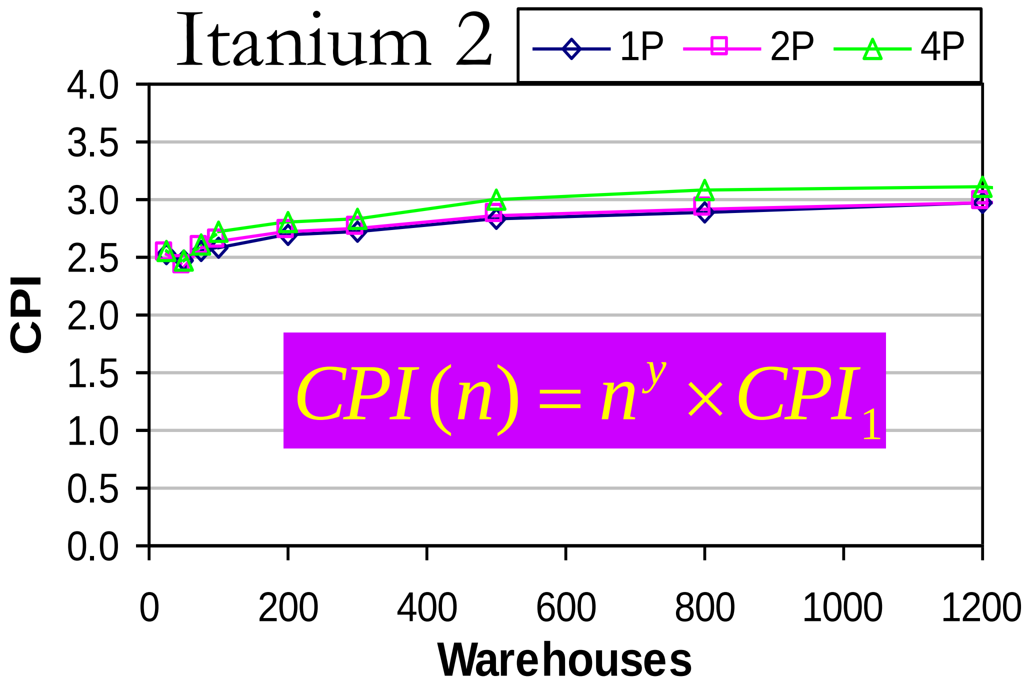
$$PL(n) = \left(n^x \times PL_1\right) \Big| \, 1.0 \ge x \ge 0.0$$

[Rich Hankins & Murali Annavaram, 2004]

CPI: Overhead Scaling

$$TPS = \frac{n \times Frequency}{IPX(n) \times CPI(n)}$$





- CPI increases with a "knee" but less sharp on Itanium 2 !
- Overall CPI trend strongly determined by user CPI
  - User mode execution time more than 90% on IPF, 80% on Xeon
- Xeon CPI grows with P, Itanium 2 CPI does not
  - Growth attributed to higher bus utilization on Xeon

- CPI does increase with increased data set size
- CPI(n) also increases with increased n (esp. for Xeon)

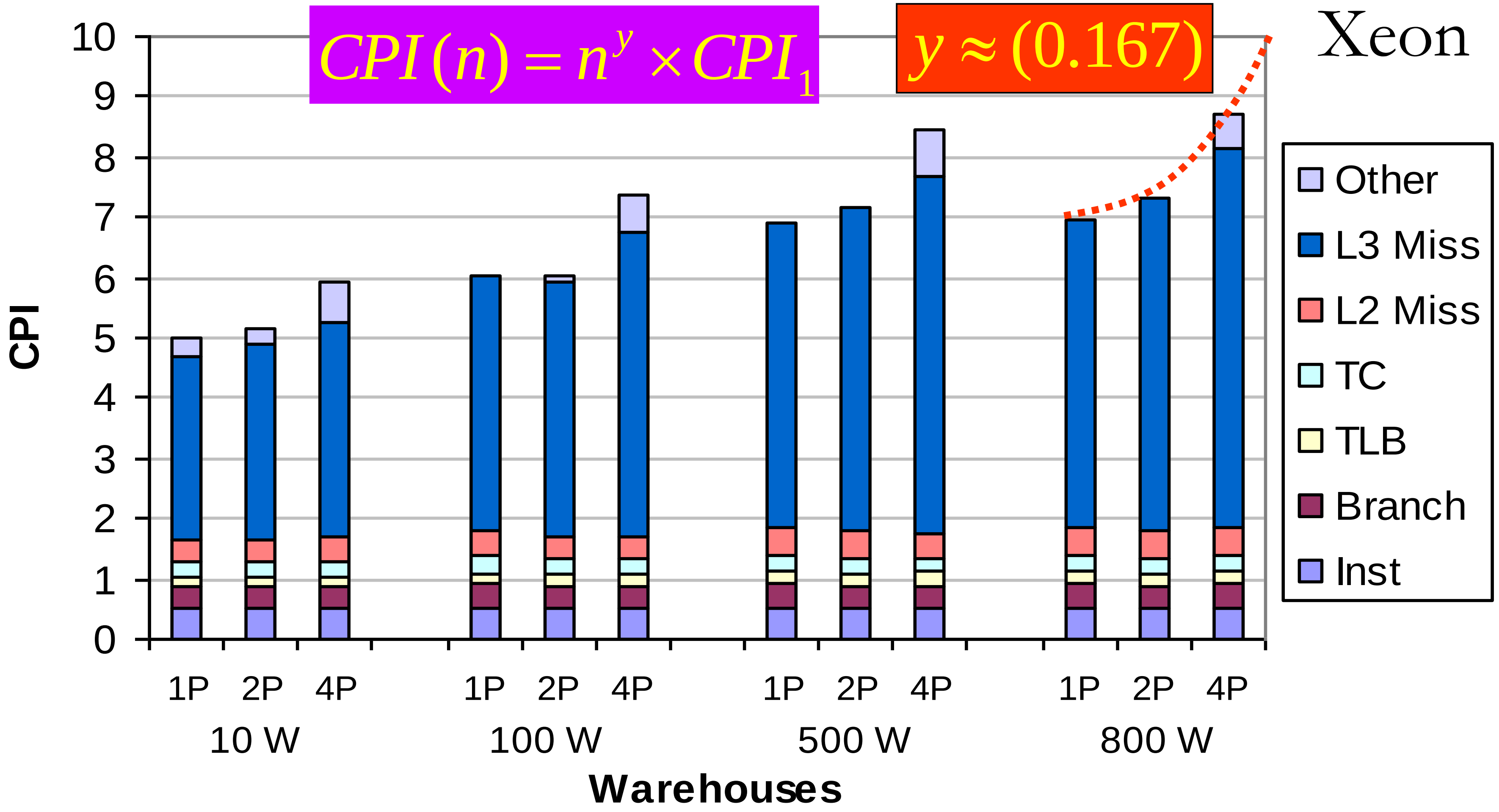
[Rich Hankins & Murali Annavaram, 2004]

$$TPS = \frac{n \times Frequency}{IPX(n) \times CPI(n)}$$

CPI Breakdown: Xeon

$$CPI(n) = n^y \times CPI_1$$

$$y \approx (0.167)$$



$$TPS = \frac{n \times Frequency}{\left(n^0 \times IPX_1\right) \times \left(n^{0.167} \times CPI_1\right)}$$

$$Speedup(n)_{MC} = \left(\frac{n}{n^x \times n^y}\right) = n^{0.833}$$

[Law #3]

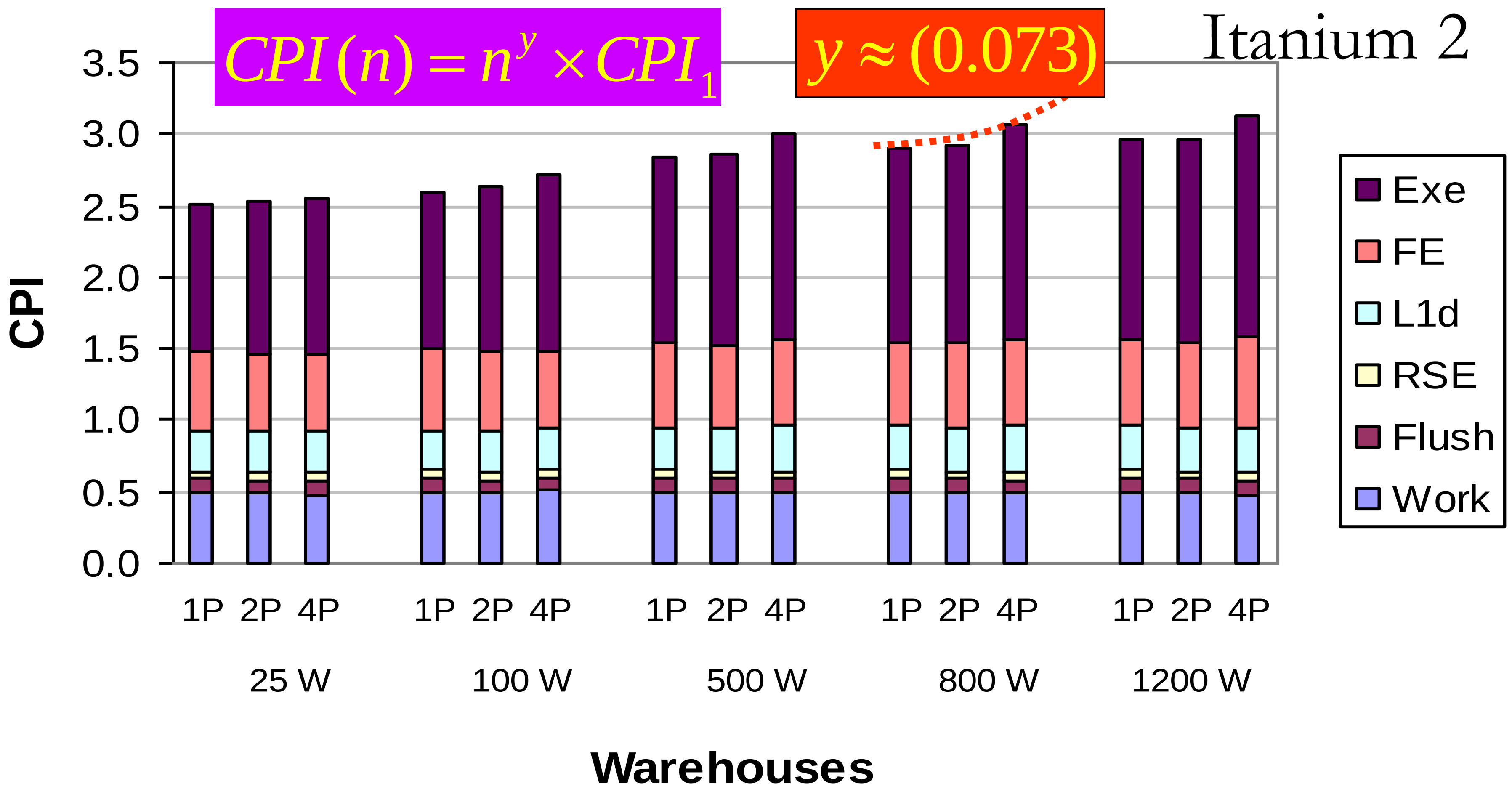
[Rich Hankins & Murali Annavaram, 2004]

$$TPS = \frac{n \times Frequency}{IPX(n) \times CPI(n)}$$

CPI Breakdown: Itanium 2

$$CPI(n) = n^y \times CPI_1$$

$$y \approx (0.073)$$



$$TPS = \frac{n \times Frequency}{(n^0 \times IPX_1) \times (n^{0.073} \times CPI_1)}$$

$$Speedup(n)_{MC} = \left(\frac{n}{n^x \times n^y} \right) = n^{0.927}$$

[Law #3]

CPI Contributions

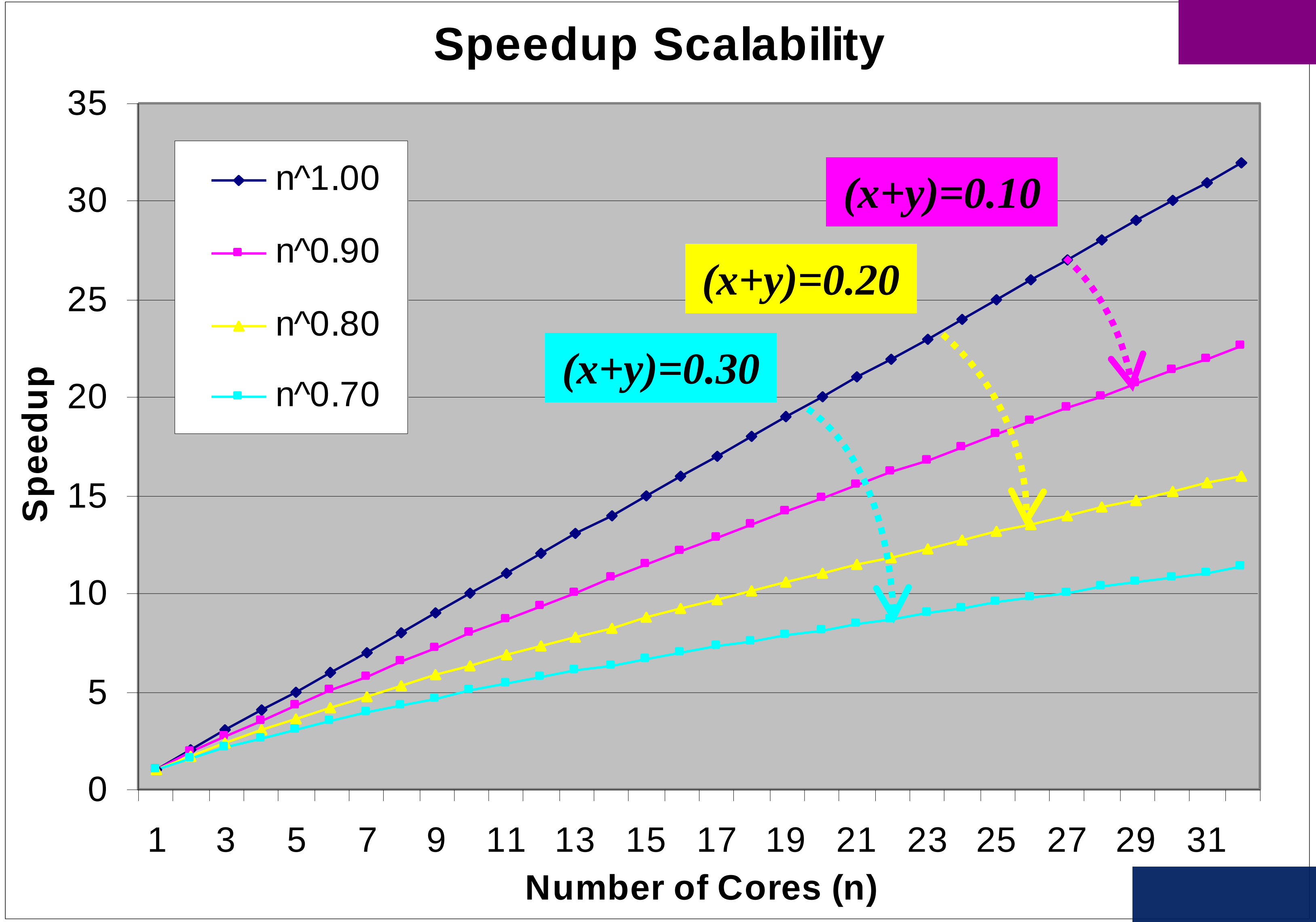
$$TPS = \frac{n \times Frequency}{IPX(n) \times CPI(n)}$$

$$CPI(n) \begin{cases} CPU & \begin{cases} Execute \\ Dep.\ Stalls \\ Resource \end{cases} \\ Cache & \begin{cases} Cache/TLB \\ Coherence \end{cases} \\ Mem & \begin{cases} DRAM \\ Bus \end{cases} \\ MP & \begin{cases} Interconnect \\ Synchronization \end{cases} \end{cases}$$

$$CPI(n) = \left(n^{y} \times CPI_{1}\right) \mid 1.0 \geq y \geq 0.0$$

MC/MP Scalability Impedance

$$Perf_{MC} = \frac{n \times Frequency}{\left(n^x \times PL_1\right) \times \left(n^y \times CPI_1\right)}$$



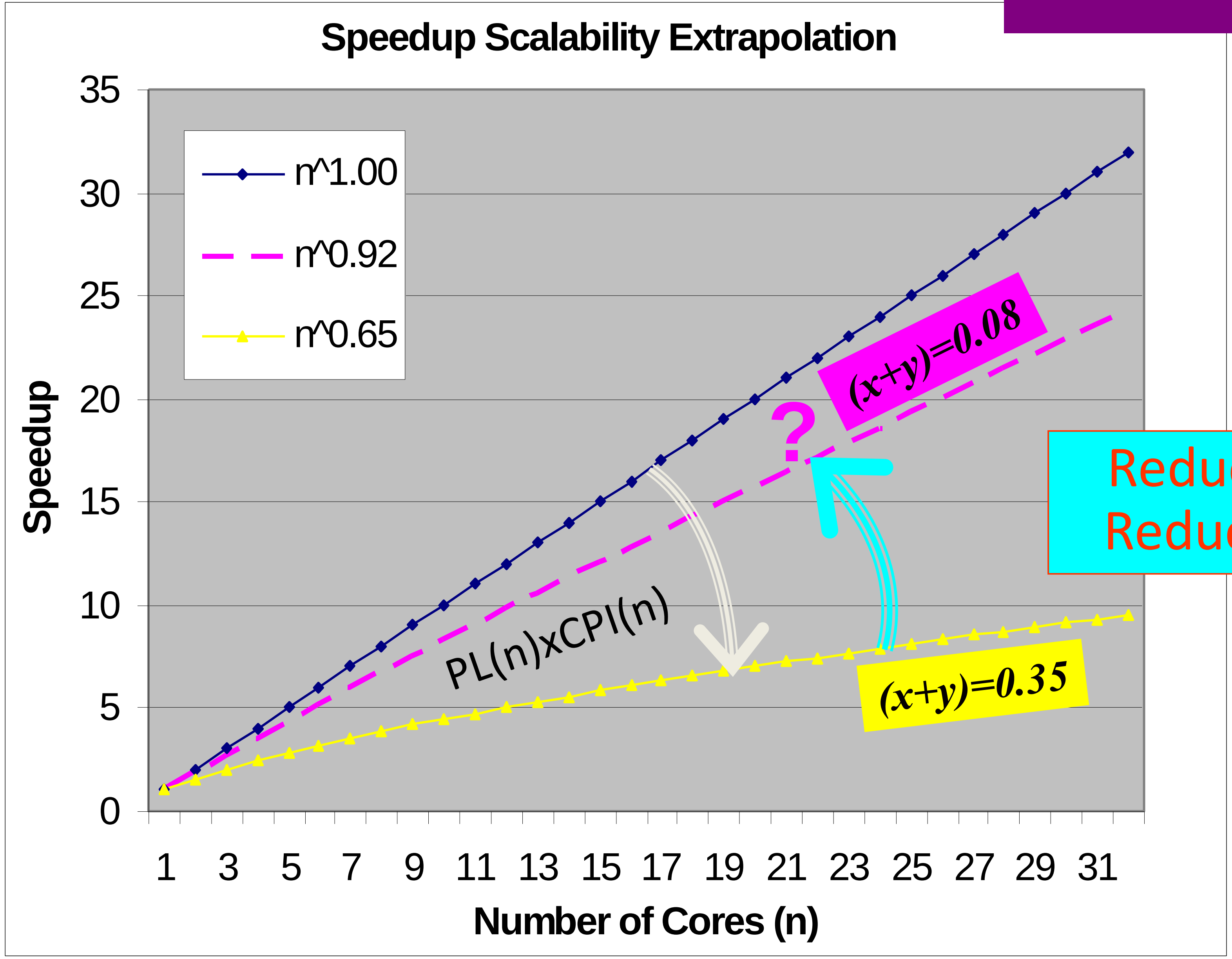
[Carole Dulong et al., 2005]

1p-16p scaling	1-(x+y)	R^2
SEMPHY	**0.993**	0.999
PLSA	**0.963**	0.999
Rsearch	**0.931**	0.997
SVM-RFE	**0.786**	0.970
SNPs	**0.685**	0.967
GeneNet	**0.642**	0.983

$$Speedup(n)_{MC} = \left(\frac{n}{n^x \times n^y} \right) = n^{1-(x+y)}$$

MC/MP Scalability Headroom

$$Perf_{MC} = \frac{n \times Frequency}{\left(n^{x} \times PL_{1}\right) \times \left(n^{y} \times CPI_{1}\right)}$$



Reduce PL(n)
Reduce CPI(n)

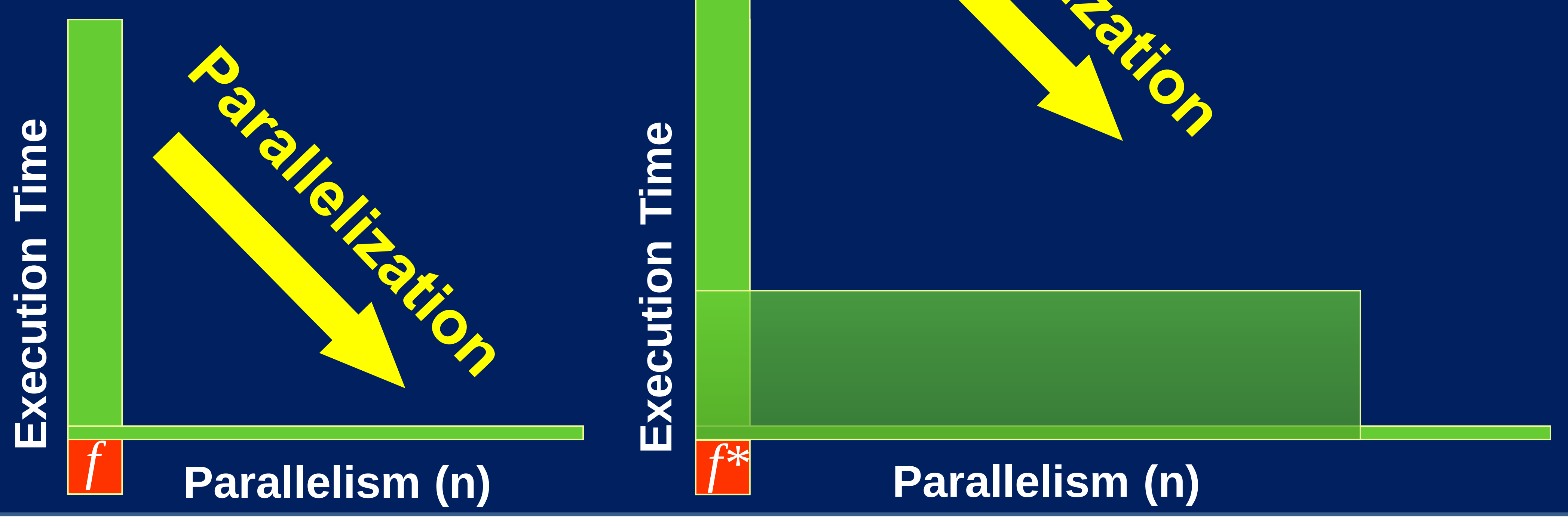
Conspiring Forces Against MC/MP Performance Scaling: Three Forms of Scalability Impedances

- **Algorithm**
  - Limitation of Language and Algorithm
  - Tyranny of Amdahl's Law (sequential bottleneck)
- **Architecture**
  - Increase of Path-Length Undermines Scalability
  - Increase of CPI also Undermines Scalability
- **Power/Thermal**
  - Increased Complexity and Inefficiency of Design
  - Super-linear Power Scaling Relative to Performance

Law #4 – Algorithm and Performance
(Amdahl's Law & Gustafson's Law)

$$Speedup(n)_{ALG} = \frac{1}{\left(\frac{f}{1}\right) + \left(\frac{(1-f)}{n}\right)}$$

f = sequential %



Two Distinct & Correlated Dimensions of Performance Scalability (Impedance)

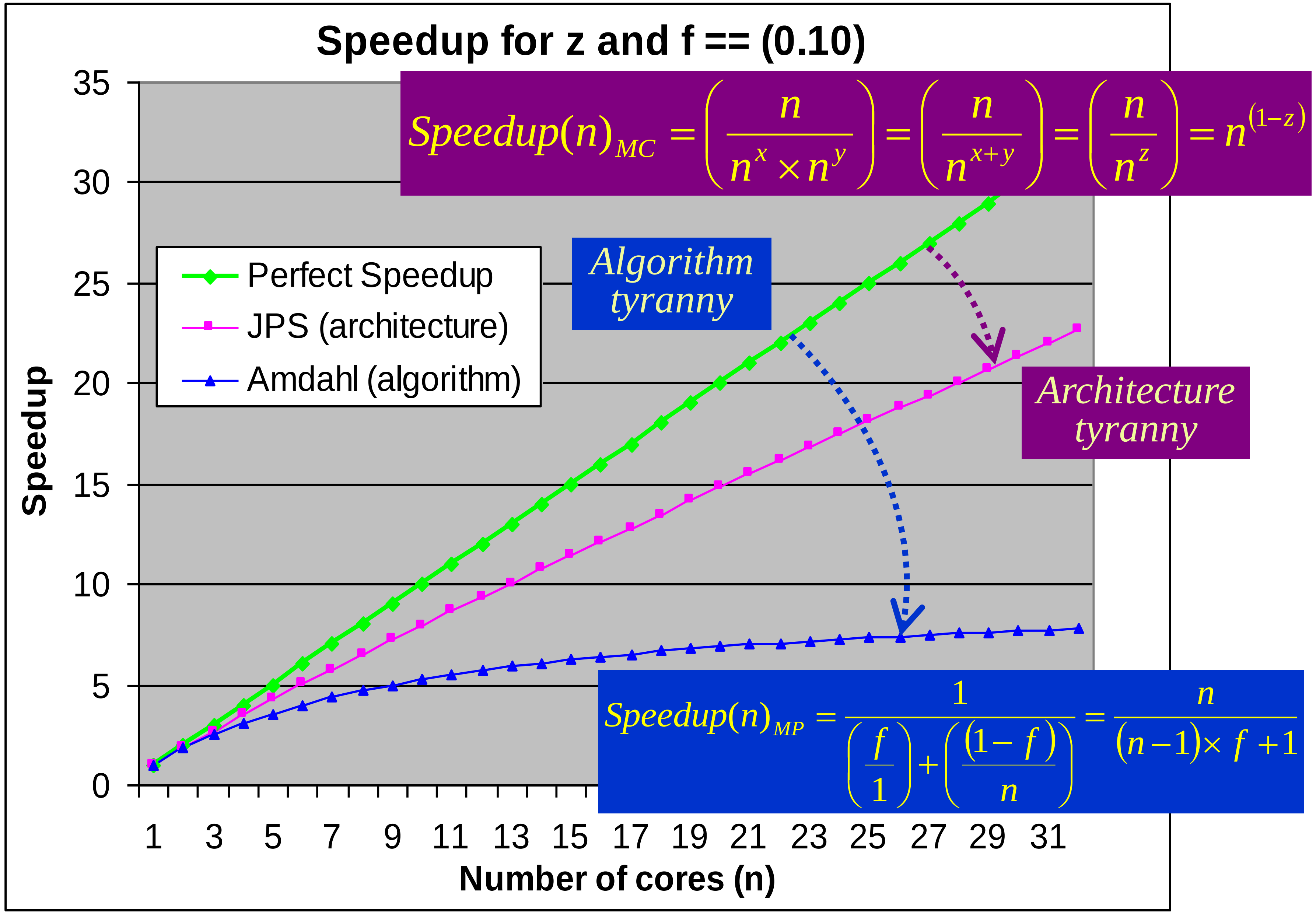
Algorithm Scalability: (Amdahl's Law)

$$Speedup(n)_{ALG} = \frac{1}{\left(\frac{f}{1}\right)+\left(\frac{(1-f)}{n}\right)} = \frac{n}{(n-1)\times f+1}$$

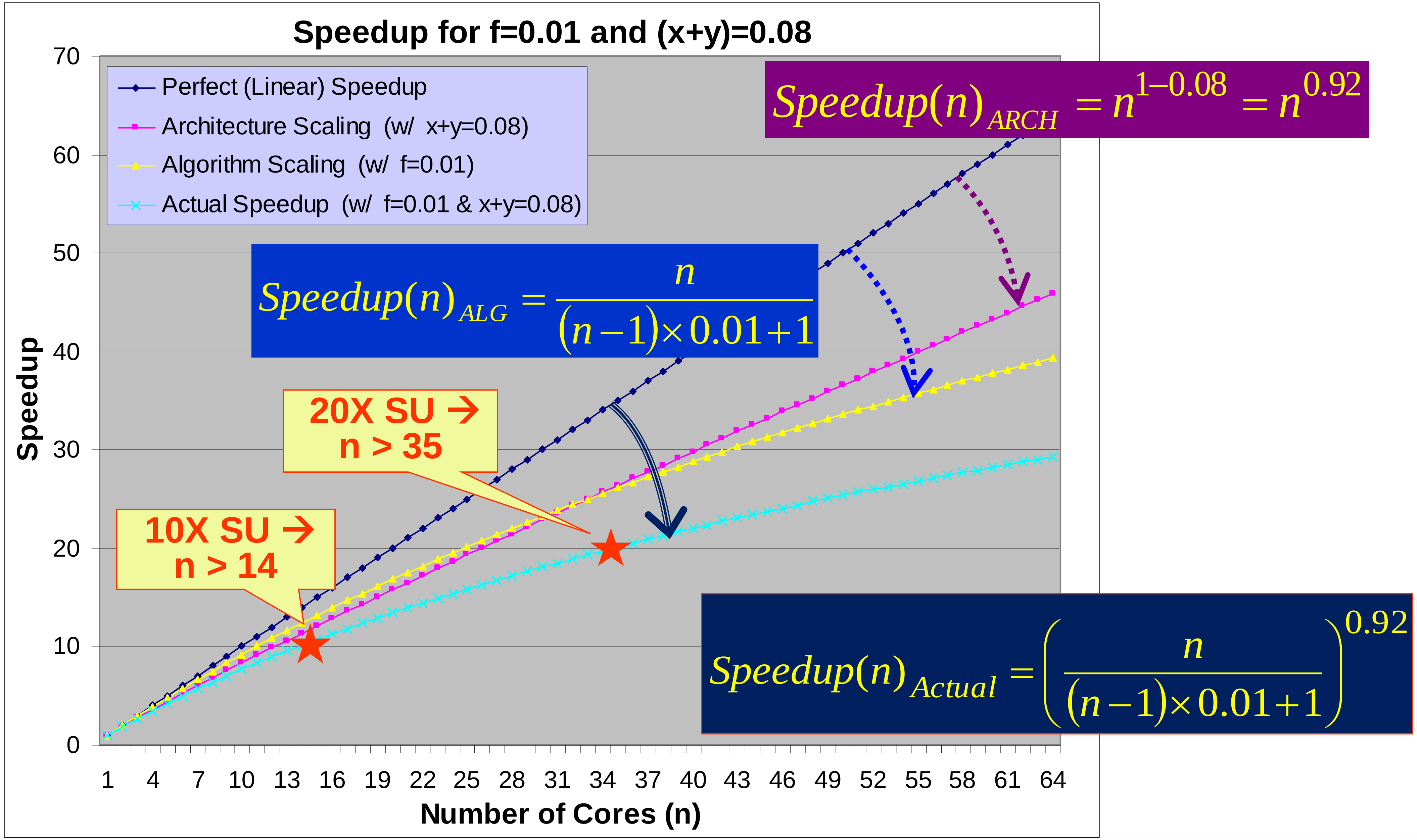
Architecture Scalability:

$$Speedup(n)_{ARCH} = \left(\frac{n}{n^x \times n^y}\right) = \left(\frac{n}{n^{x+y}}\right) = n^{1-(x+y)}$$

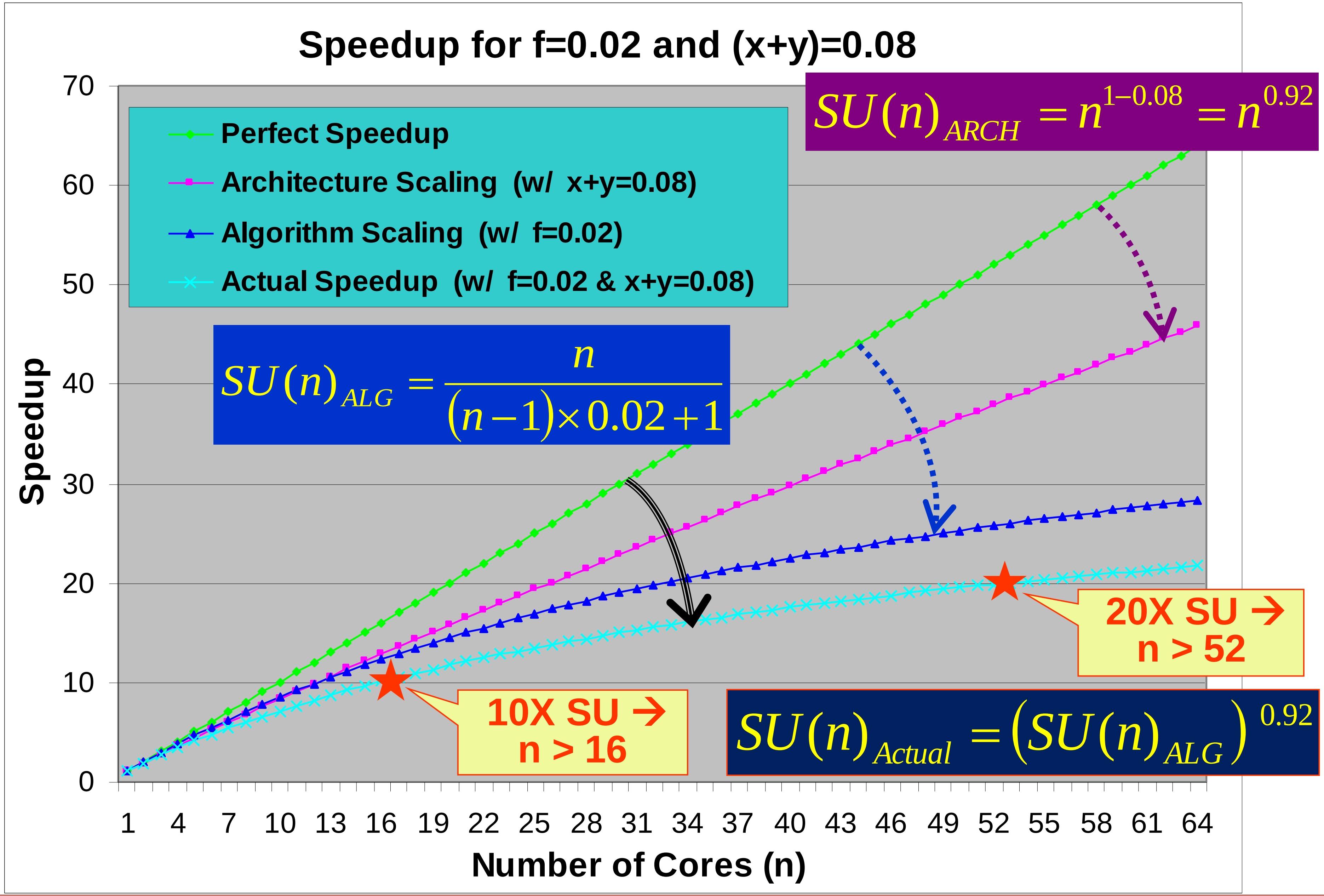
Relative (Additive) Degrees of Tyranny



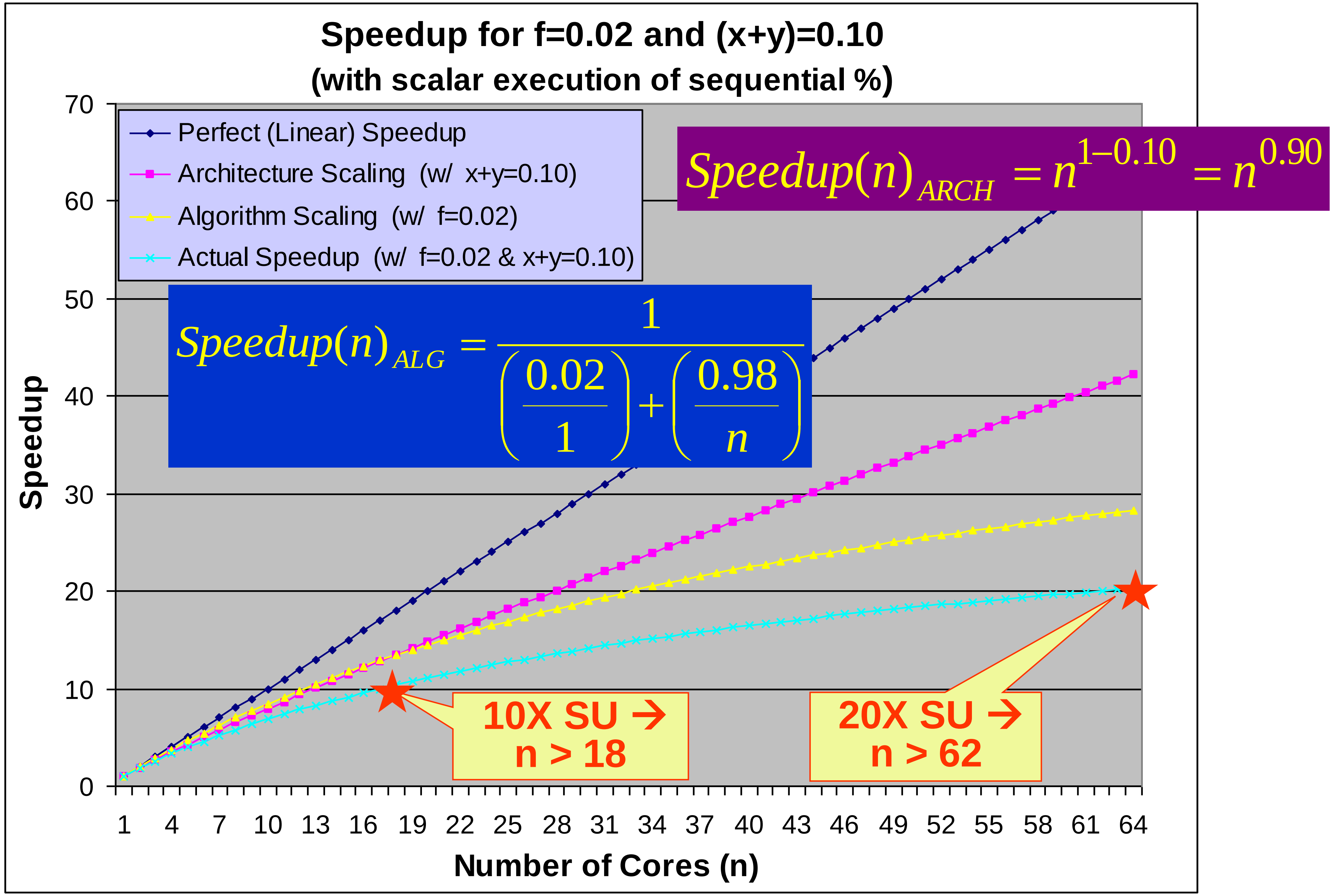
Combined Effect on Actual Speedup (f=0.01, x+y=0.08)



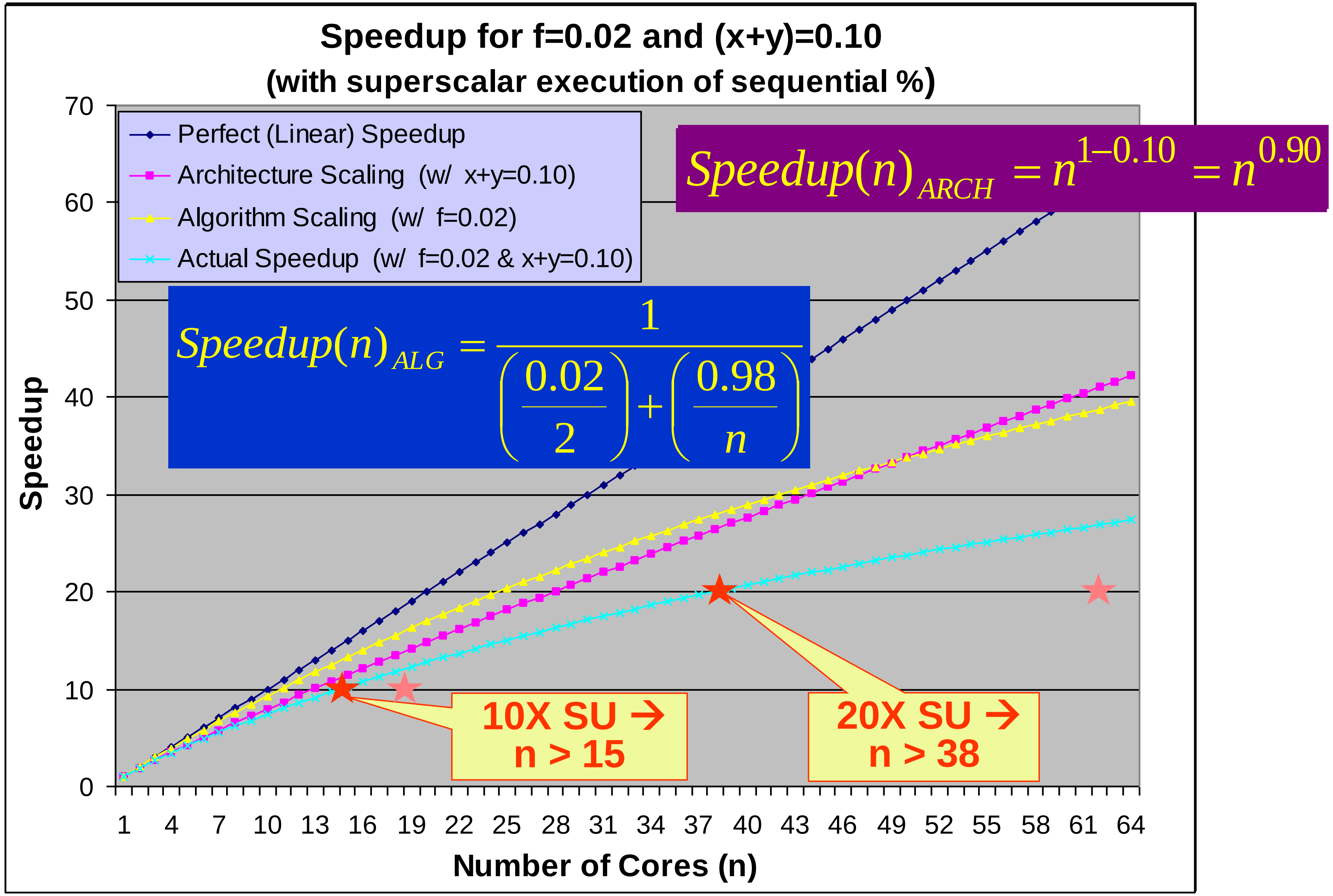
Combined Effect on Actual Speedup (f=0.02, x+y=0.08)



Combined Effect on Actual Speedup (f=0.02, x+y=0.10)



Impact of Latency Performance on MP Performance



Law #5 – Power and Performance

$$Watt = \frac{Joule}{second} = \frac{Joule}{instruction} \times \frac{instruction}{cycle} \times \frac{cycle}{second}$$

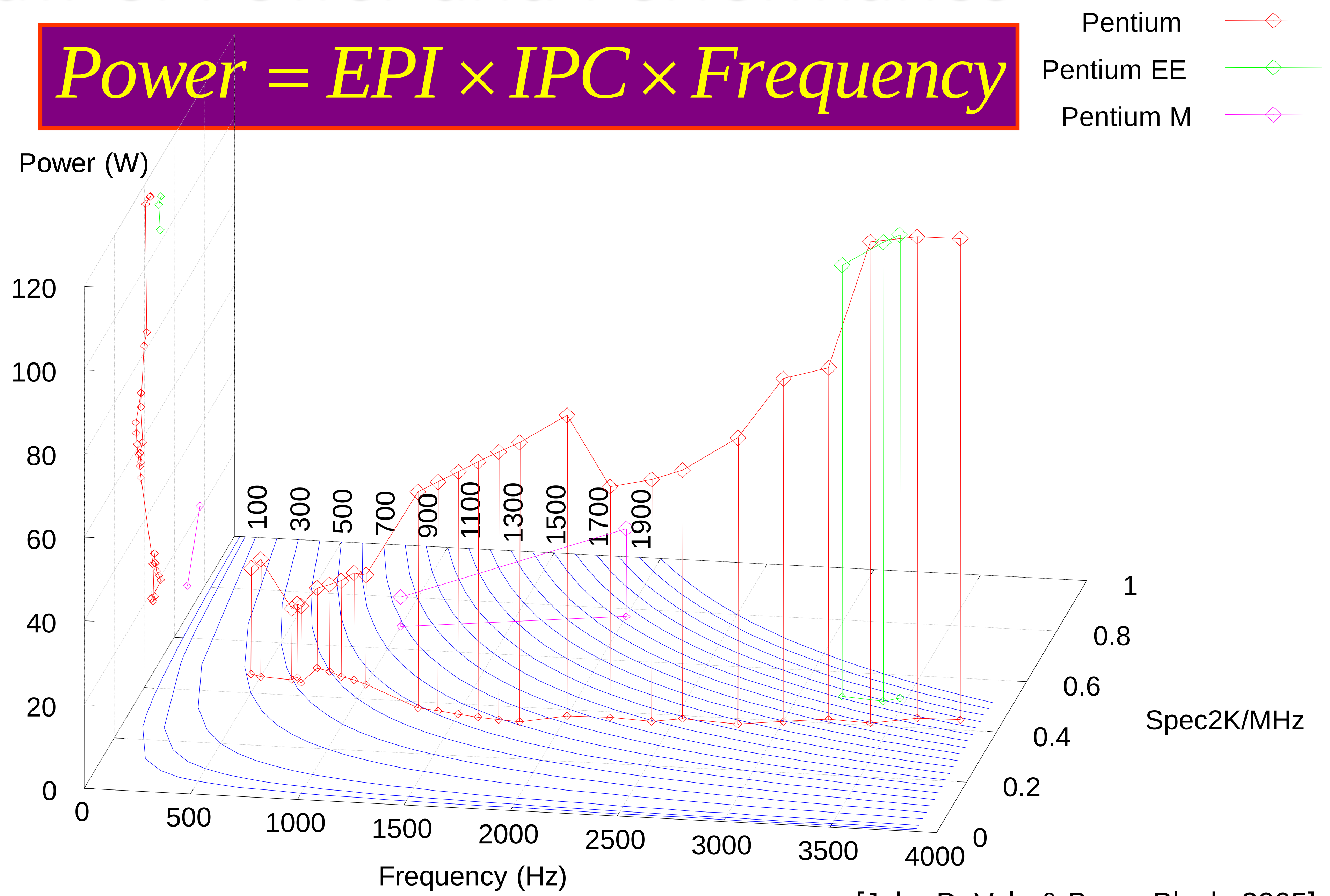
$$Power = \boxed{EPI \times IPC \times Frequency}$$

$$Performance = \frac{Frequency}{PathLength \times CPI} = \frac{IPC \times Frequency}{PathLength}$$

$$Power = \boxed{EPI \times Performance \times PathLength}$$

Iron Law of Power and Performance

$$Power = EPI \times IPC \times Frequency$$

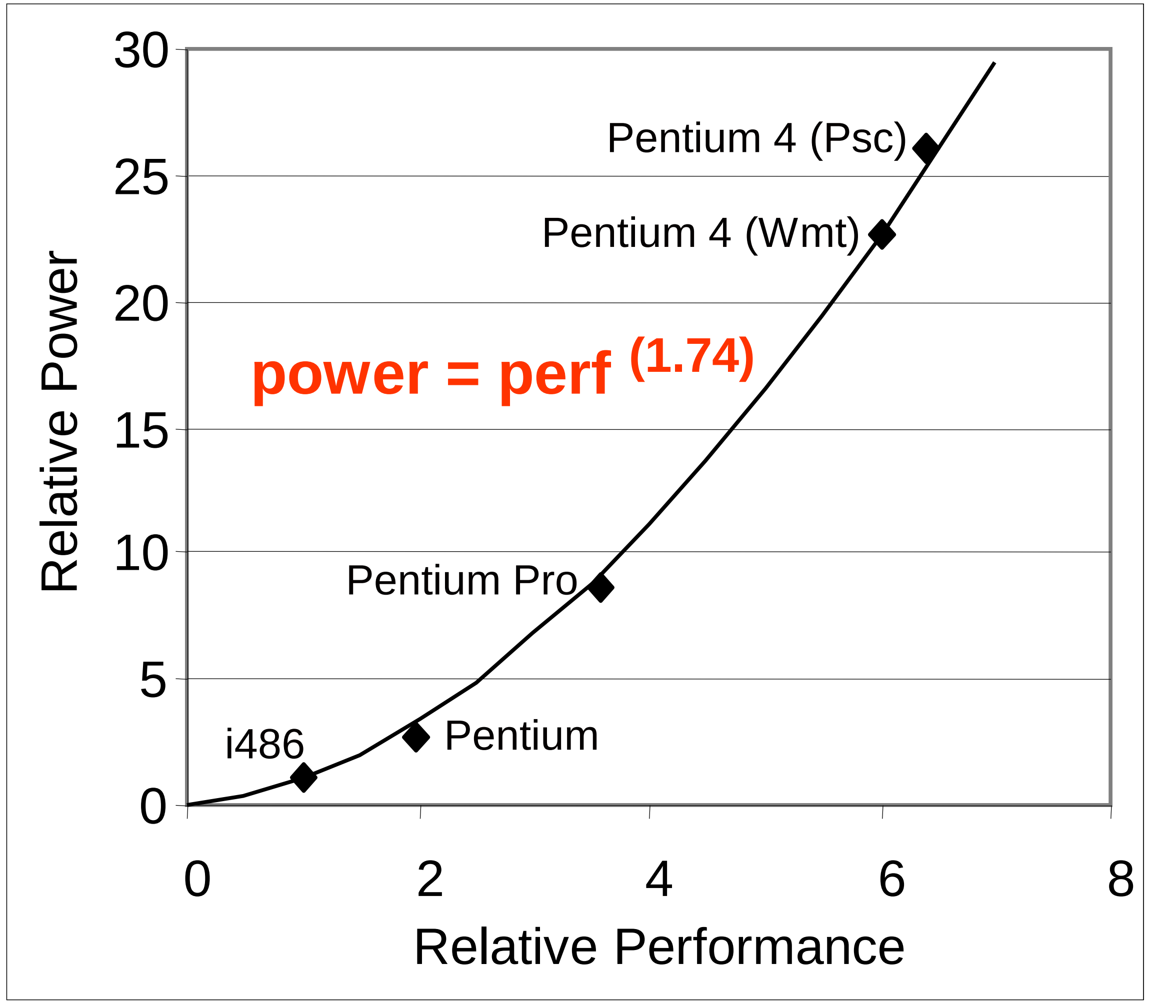


[John DeVale & Bryan Black, 2005]

[Ed Grochowski, 2004]

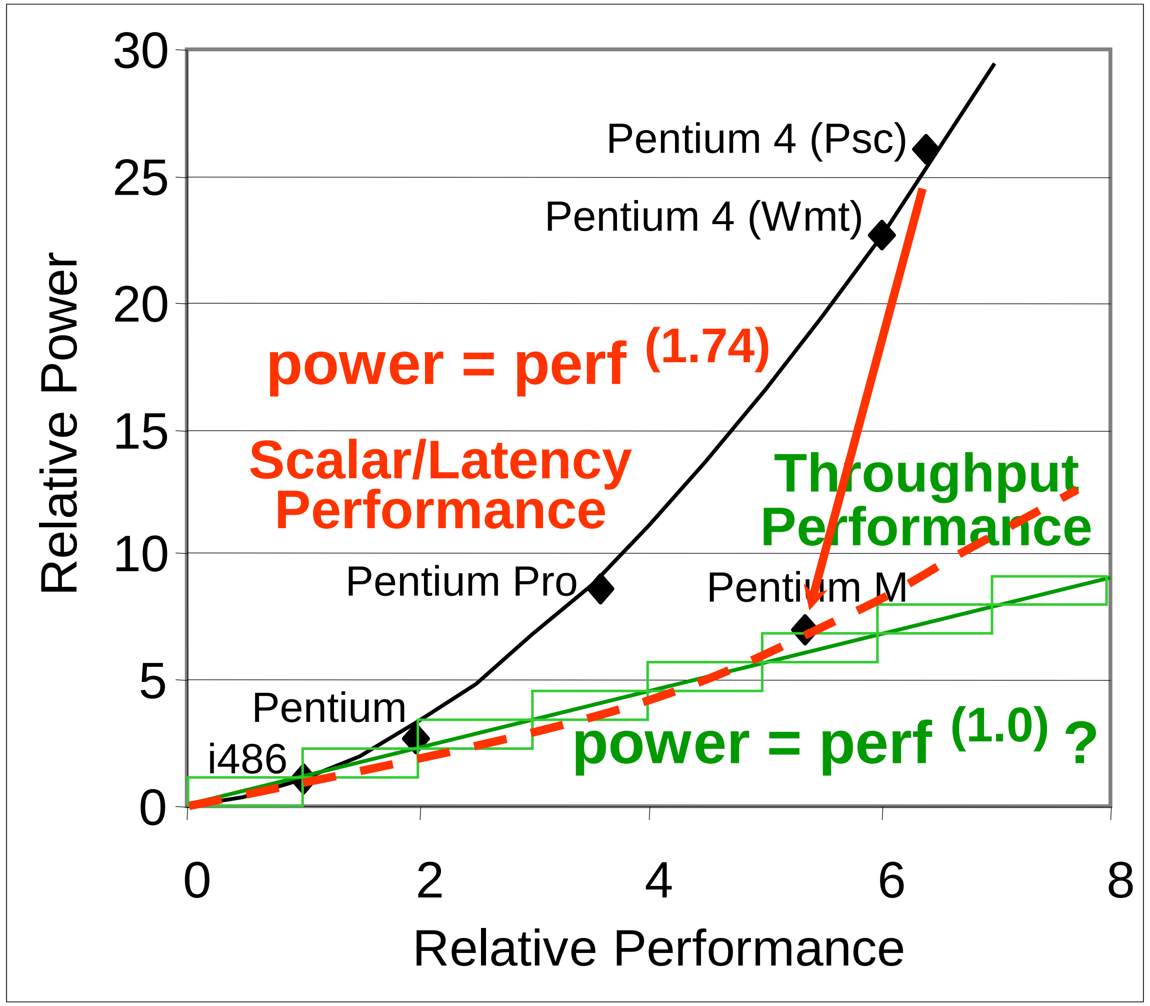
Power and Scalar (Latency) Performance

- For comparison
  - Factor out contributions due to process technology
  - Keep contributions due to microarchitecture design
  - Normalize to i486™ processor
- Relative to i486™ Pentium® 4 (Wmt) processor is
  - 6x faster (2X IPC at 3X frequency)
  - 23x higher power
  - Spending 4 units of power for every 1 unit of scalar performance



Power and Latency/Throughput Performance

[Ed Grochowski, 2005]



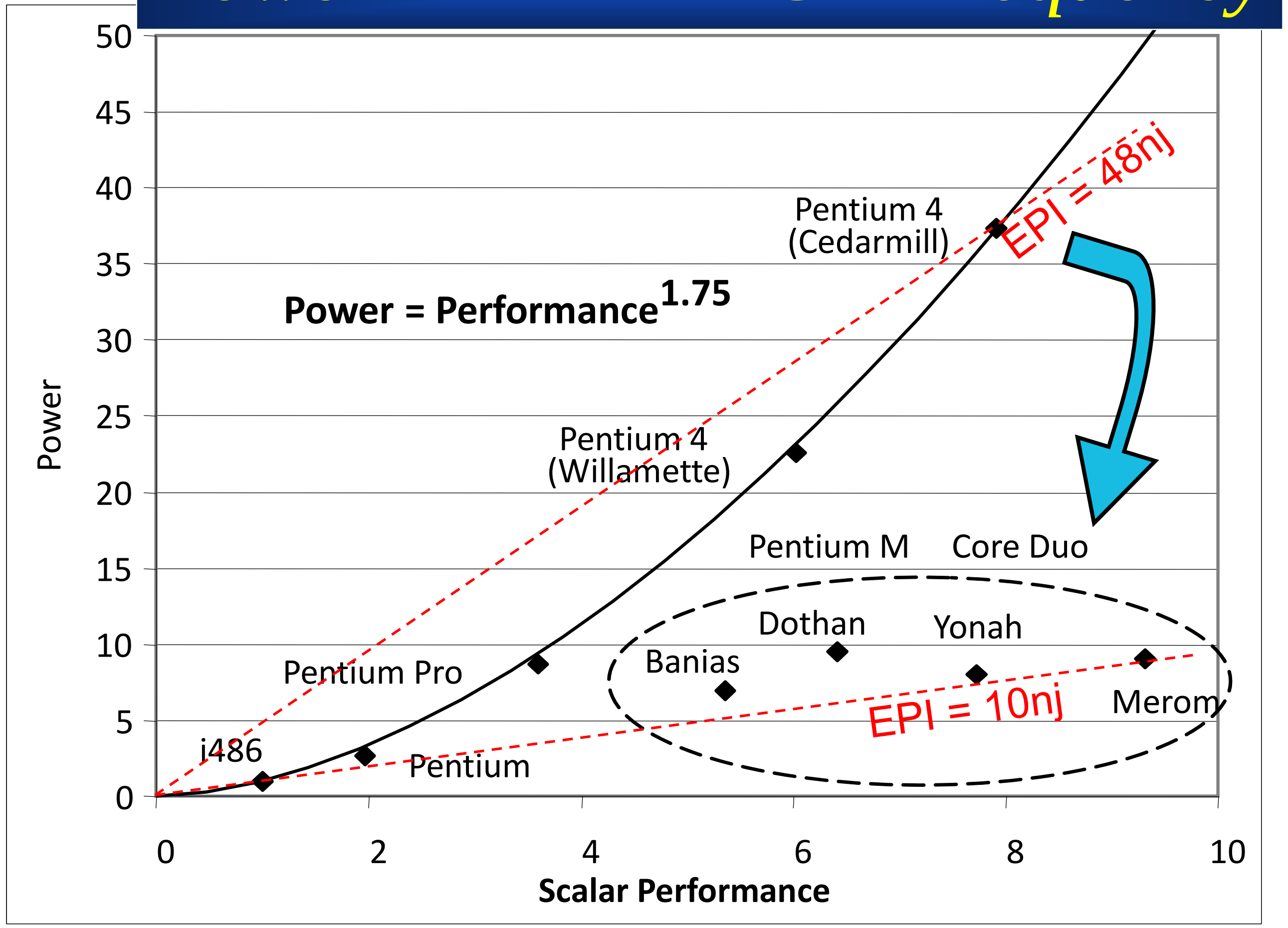
CPU	EPI
I486	7 nj
P5	10 nj
P6	17 nj
P4P-wmt	27 nj
P4P-psc	29 nj
Pentium M	9 nj

Low EPI

- *Assume a large-scale CMP with potentially many cores*
- *Replication of cores results in (nearly) proportional increases to both throughput performance and power (hopefully).*

Power/Performance (EPI) Evolution

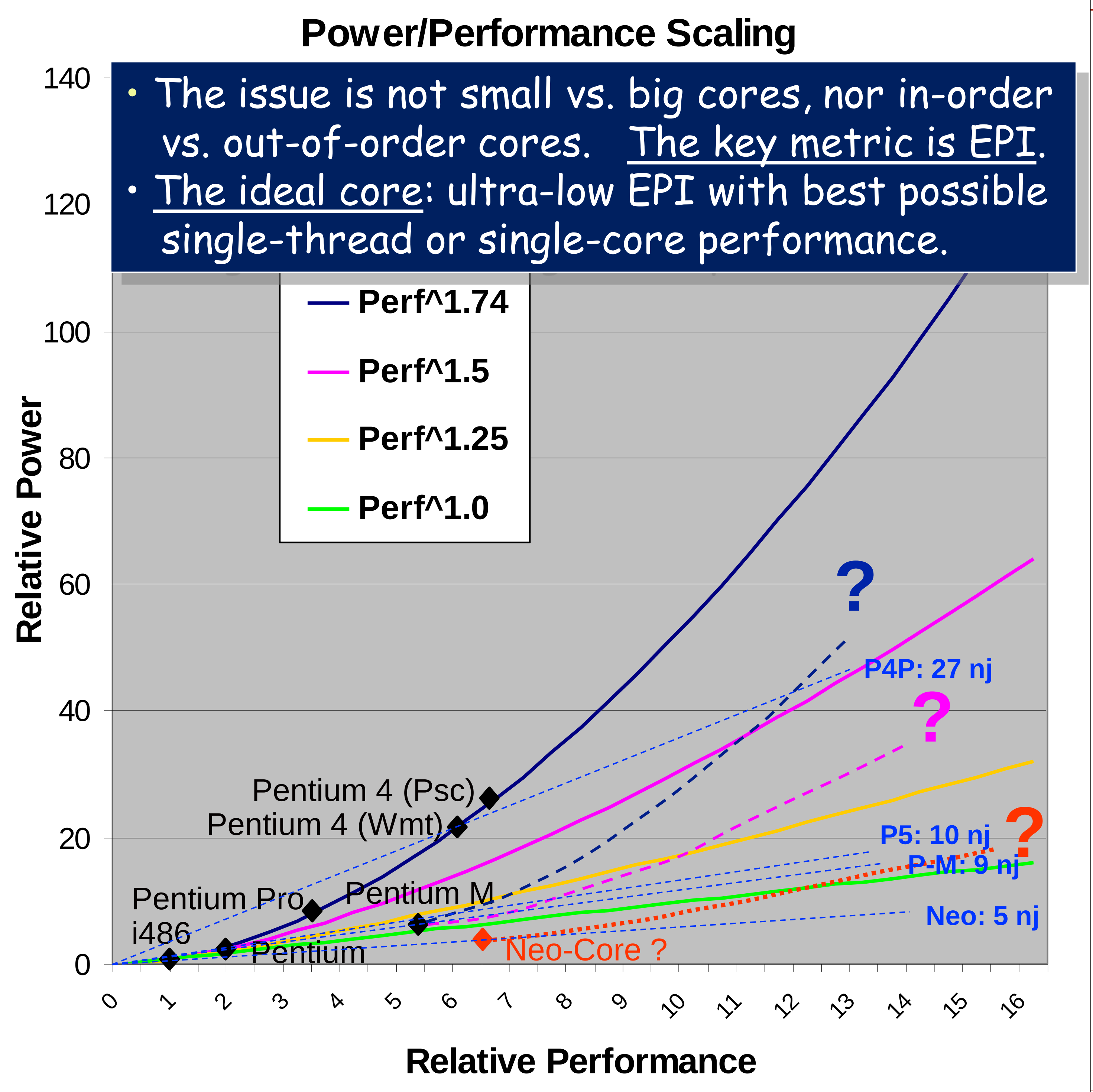
$$Power = EPI \times IPC \times Frequency$$



[Ed Grochowski, 2004]

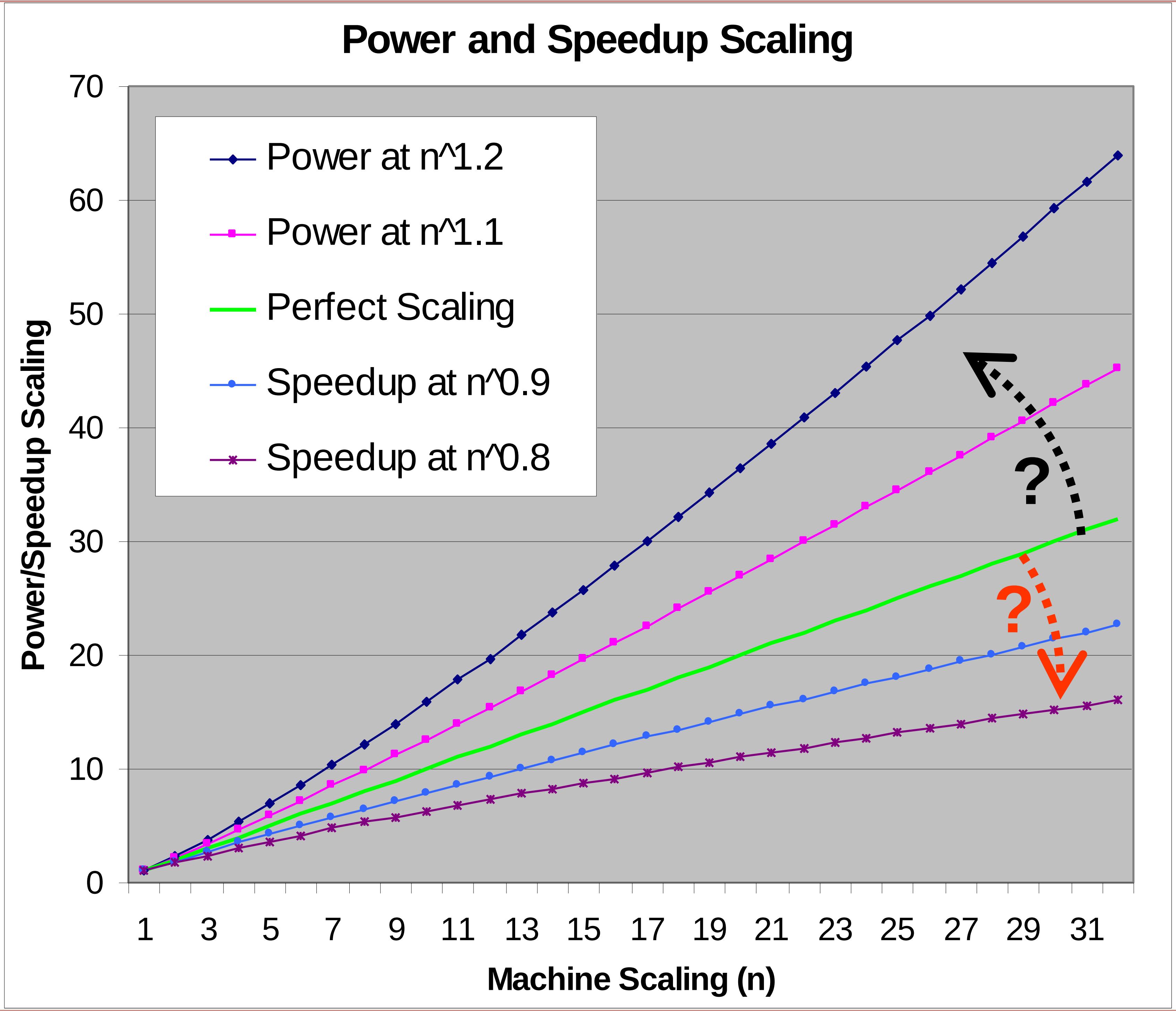
Intel Microprocessors	EPI (nj) 65nm at 1.33v
i486	10
Pentium	14
Pentium Pro	24
Pentium 4 (WMT)	38
Pentium 4 (CDM)	48
Pentium M (Banias)	13
Pentium M (Dothan)	15
Core Duo (Yonah)	11
Core Duo (Merom)	10

- Power: single core power (relative to i486 baseline)
- Performance: SPECint performance (relative to i486 baseline)
- EPI: average energy spent per instruction (in nano-joules)



CPU	EPI	SU
i486	7 nj	1
P5	10 nj	2
P6	17 nj	3.5
P4P-wmt	27 nj	6
P4P-psc	29 nj	6.5
Pentium M	9 nj	5.5
Neo-Core	5 nj	6.5

NP/DSP/GPU	EPI
IXP2800	1 nj
TMS320C6713	0.7 nj
GeF7800GTX	0.6 nj

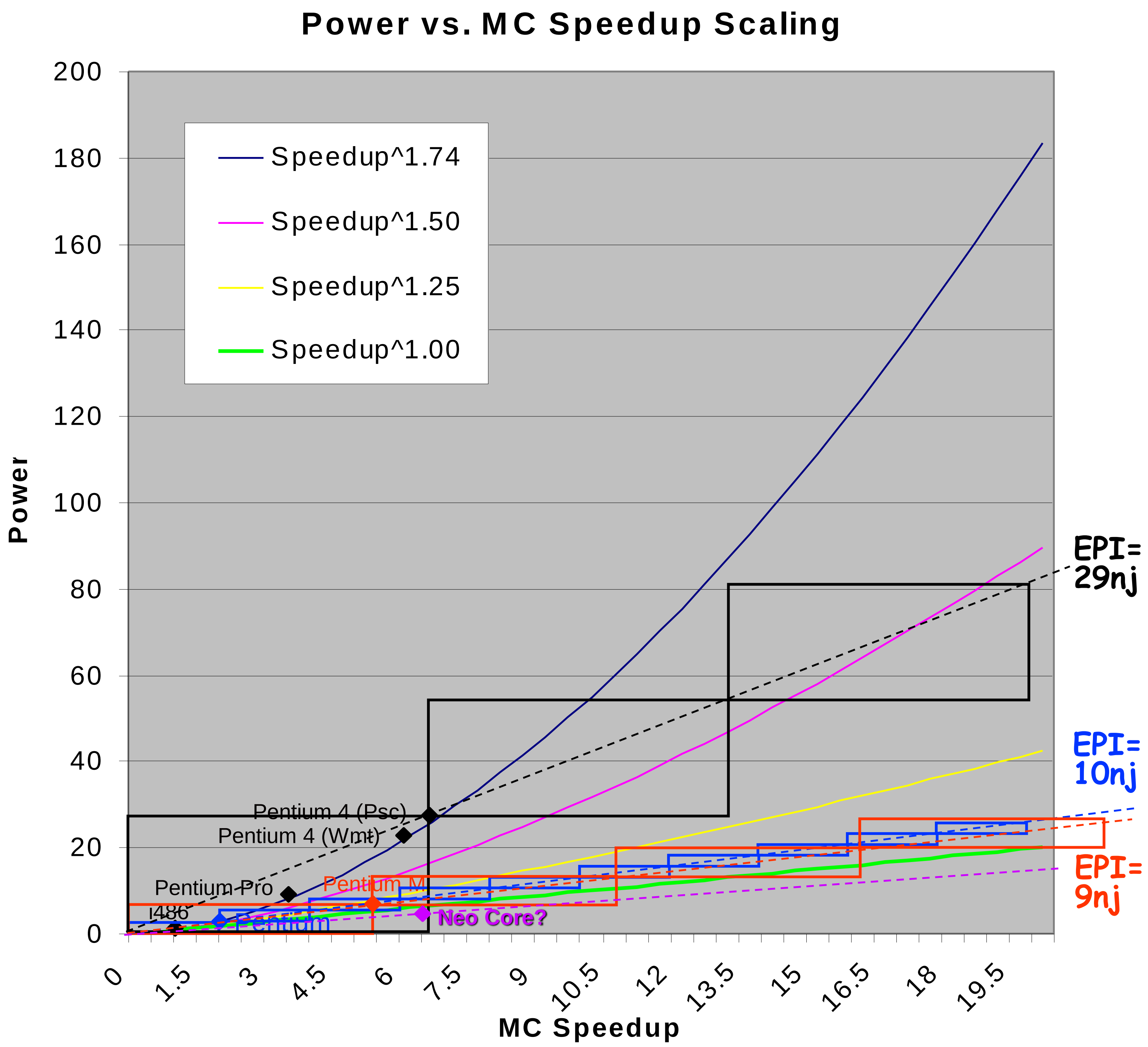


<u>Power scaling challenges:</u>

- Low EPI cores
- Un-core scaling

<u>Speedup scaling challenges:</u>

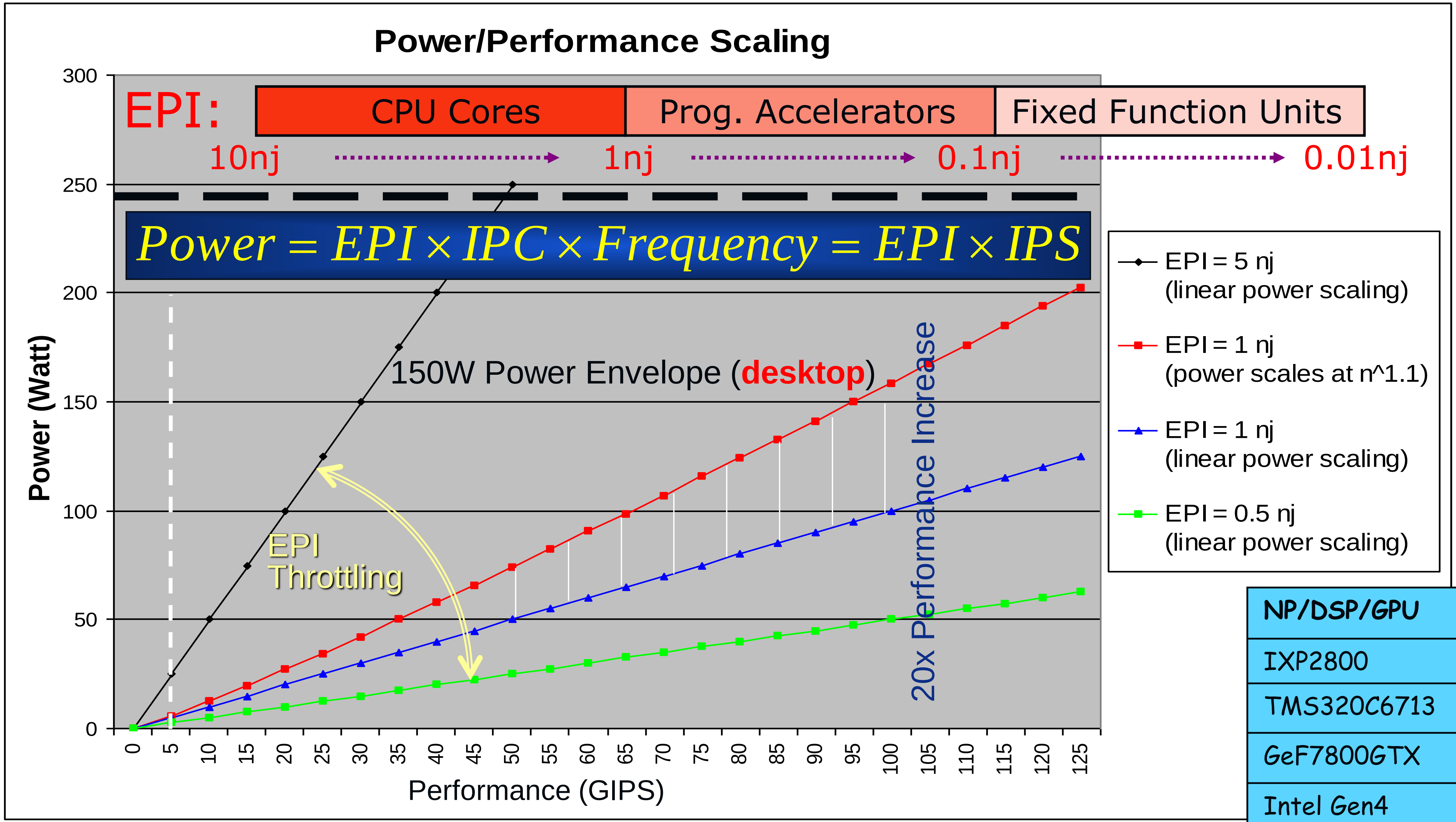
- Algorithm
  - Sequential %
- Architecture
  - PL scaling
  - CPI scaling



CPU	EPI	SU
i486	7 nj	1
P5	10 nj	2
P6	17 nj	3.5
P4P-wmt	27 nj	6
P4P-psc	29 nj	6.5
Pentium M	9 nj	5.5
Neo core?	< 5 nj	6.5

- The scaling goal is not just the number of cores but <u>maximum throughput</u> within fixed power envelope.
- The key issue is not the power scaling of replicated cores, but the <u>un-core power scaling</u> that may push total power scaling towards the square law again.

Dynamic EPI Optimization for MC/MP Architectures



NP/DSP/GPU	EPI
IXP2800	~1 nj
TMS320C6713	~0.7 nj
GeF7800GTX	~0.6 nj
Intel Gen4	~0.3 nj

Quo Vadis?

- **Scalability Strategies:**
  - Algorithm – Languages & Specialized Parallelism
  - Architecture – CPI and Path Length Reduction
  - Power/Thermal – EPI Reduction & Scalable Un-core
- Power/Energy is the new scalability wall
- **Research Challenges:**
  - Sequential % mitigation for compelling workloads
  - Ultra-low EPI core with great latency performance
  - Un-core fabric with near-linear power scaling
- Un-core scaling is the new power goblin

Useful Chart on Power, Voltage, Resistance, Current

Formula 1 – Electrical power equation:

Power $P = I \times V = R \times I^2 = V^2 / R$

where power P is in watts, voltage V is in volts and current I is in amperes (DC).

If there is AC, look also at the power factor $PF = \cos \varphi$ and

φ = power factor angle

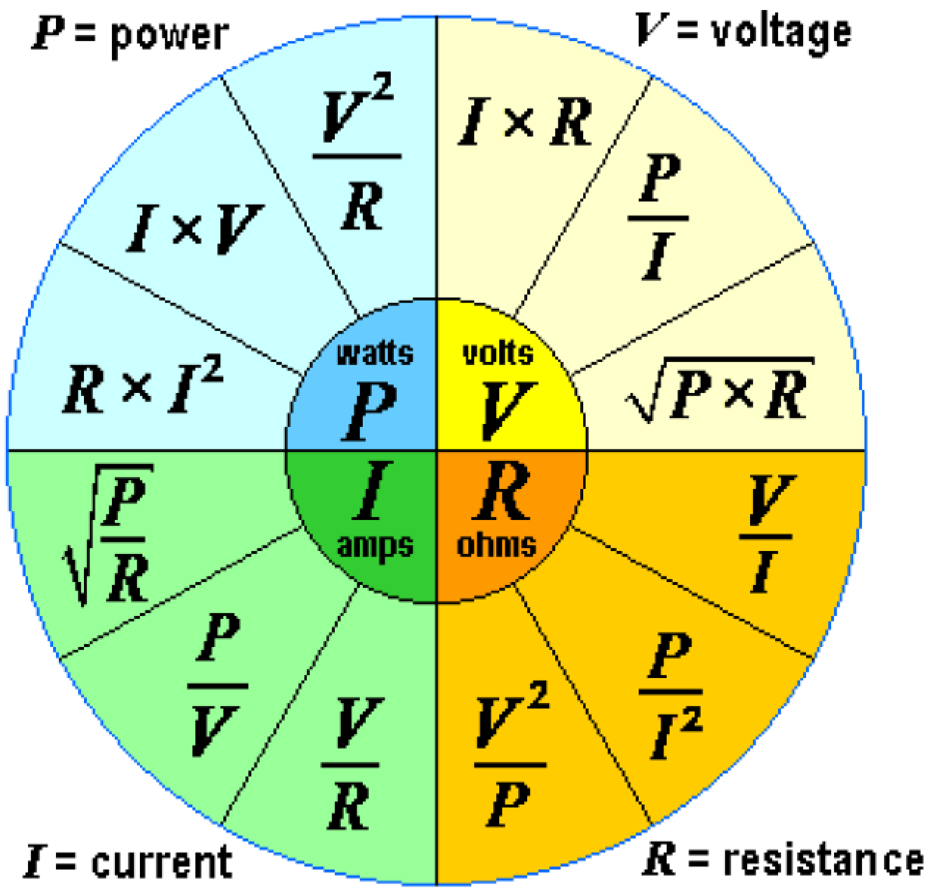
(phase angle) between voltage and amperage.

Formula 2 – Mechanical power equation:

Power $P = E / t = W / t$

where power P is in watts, **Energy** E is in joules, and time t is in seconds. 1 W = 1 J/s.

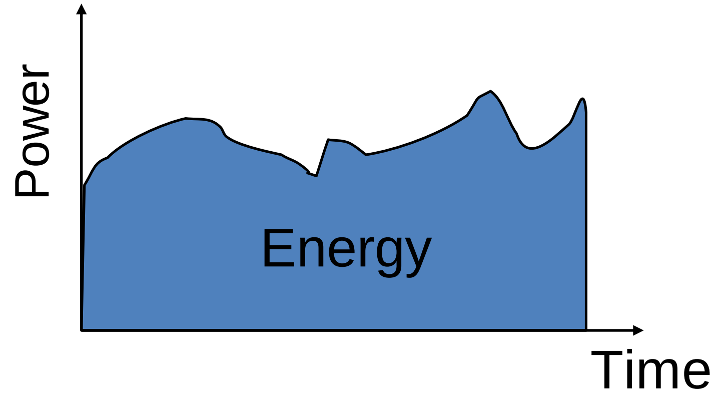
Electric **Energy** is $E = P \times t$ – measured in watt-hours, or also in kWh. 1J = 1N×m = 1W×s



http://www.sengpielaudio.com/calculator-ohm.htm

Power vs. Energy

- Energy: integral of power (area under the curve)
  - Energy and power driven by different design constraints
- Power issues:
  - Power delivery (supply current @ right voltage)
  - Thermal (don't fry the chip)
  - Reliability effects (chip lifetime)
- Energy issues:
  - Limited energy capacity (battery)
  - Efficiency (work per unit energy)
- Different usage models drive tradeoffs



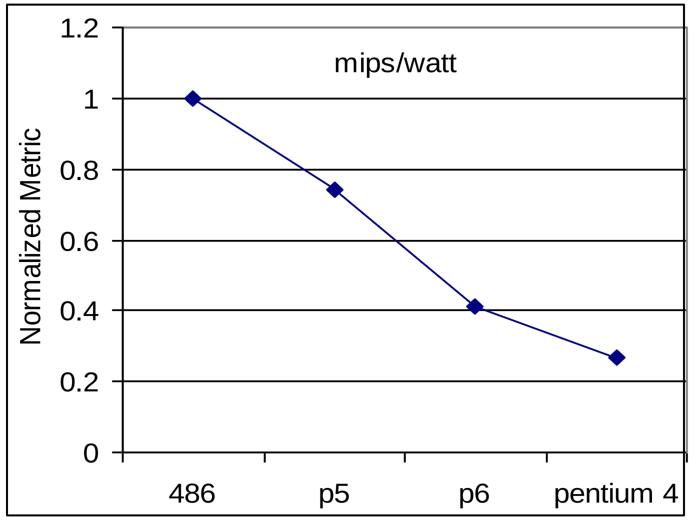
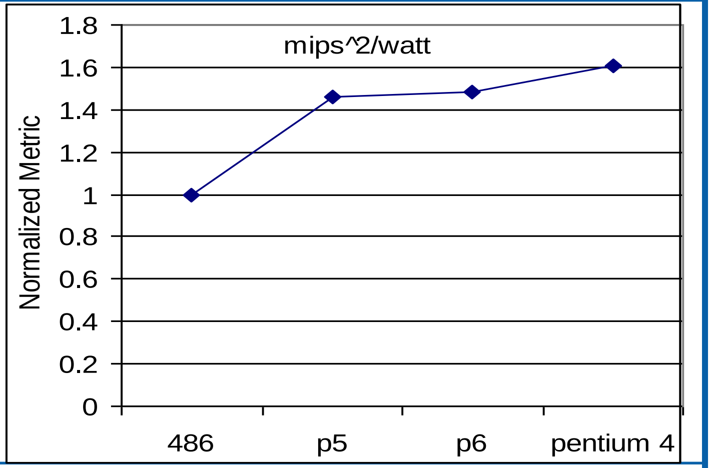
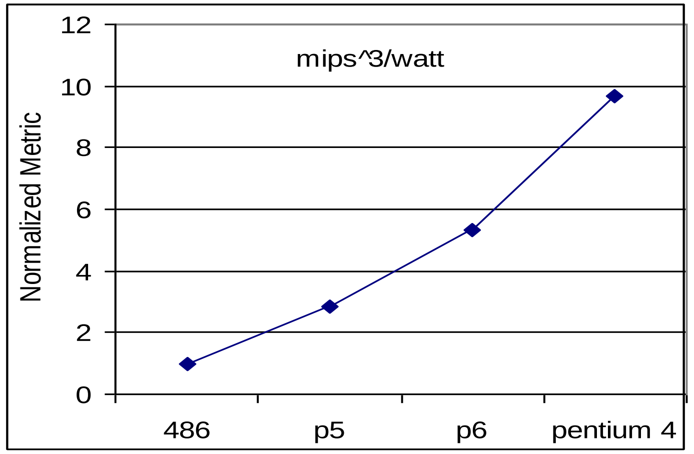
Power vs. Energy

- With constant time base, two are "equivalent"
  - 10% reduction in power => 10% reduction in energy
- Once time changes, must treat as separate metrics
  - E.g. reduce frequency to save power => reduce performance => increase time to completion => consume more energy (perhaps)
- Metric: energy-delay product per unit of work
  - Tries to capture both effects
  - Others advocate energy-delay^2
  - Best to consider all
    - Plot performance (time), energy, ed, ed^2

Performance/Power Efficiency Metrics

- Power is good metric for deciding on the thermal envelope of the processor
- Energy is good metric in battery constrained environments
  - Task executed at ½ speed but ¼ power means ½ the energy (2T * ¼ P = ½ E)
  - 2X battery life!
- Energy*Delay metric gives higher weight to performance
  - Same example above, ED ($(2T)^2$ * ¼ P) stays same
- Energy*Delay^2 gives even more weight to performance
  - Same example above shows that ½ speed is 2X worse on ED^2 metric

[Ed Grochowski, Intel, 2005]

MIPS/watt	Common measure of power efficiency Equivalent to **energy per instruction** Independent of time Ideal metric for throughput performance $\frac{\text{Mips}}{\text{Watt}} = \frac{\frac{\text{Instructions}}{\text{Second}}}{\frac{\text{Joules}}{\text{Second}}} = \frac{\text{Instructions}}{\text{Joule}}$	
MIPS^2/watt	Equivalent to **energy • delay** Common metric for comparing logic families	
MIPS^3/watt	Equivalent to **energy • delay^2** Assign increasing weight to time Appropriate metric for latency performance	

Energy Storage Challenge for Mobile Devices

How long between recharges?

iPad2
- 25Wh = 90kJ
- 10h use = avg 2.5W

Kindle3
- 6.5Wh = 23kJ
- "30 day use" = 30h
  avg 220mW

Typical smartphone
- 32g, 13cc, 5.5Wh = 18kJ
- +5h charging @ max 1W
- 20mW static power = 10 days standby
- 150mW notifications = 1.3 day standby
- "typical usage" 5kJ active + 13kJ standby =
  1 battery charge

Overarching Research Challenge for Mobile Devices

$$Max\left(\frac{UserExperience}{Power \times Time}\right)$$

<u>GOAL:</u>

Improve Energy Efficiency
by 100x in 5 years

Holistic System Approach

End-to-End System-Wide Approach:
focus on the three major subsystems

Communication: (modems)	User Interfaces: (displays)	Computation: (processors)

Unique Opportunities in the next 5 years:
"emergence of heterogeneity in all three areas"

Key Challenges…how far can we go? [Per Ljung, 2012]

Key Areas	Description
Active Power Saving	➢ **Context Based Power Management:** Offloading of communication to available connectivity, and computation to companion devices or the cloud. ➢ **Workload Based Power Management:** Manage power consumption based on actual usage and workload scenarios by leveraging heterogeneous cores and smart parallelism to reduce overall power.
Standby Power Saving	➢ **Near-Zero-Power Standby Mode:** Low-power always-on transflective bistable (TF/BS) displays; eager hibernation with instant resume. ➢ **Ultra-Low-Power Always-On Device:** Device with minimal standby functionality and seamless quick switch over to companion devices.
Energy Harvesting	➢ **Casual Charging:** Wireless inductive charging; solar charging for large surface devices. ➢ **Anticipatory Preexecution:** Speculative cross-device or cloud-based preprocessing and content prefetching.

Active Power Saving

ideal 100%

1.5h @ 900mW = 1.35Wh   5Wh battery: 30%

avg app   goal <100mW

80%

20%

➤ Context-Based Power Management

- Offload Communication (Local ULP radio)
  - 1m radio instead of 1000m cellular
  - Dongle, access point, femtocell box
- Offload Computation (Cross device)
  - Cloud/companion pre-compute/pre-render

80% time in home/office
40mW vs 1W

25x

trade cheap communication
for expensive computation
40mW vs 1W

25x

➤ Workload-Based Power Management

- Power consumption based on usage scenario
  - Activity, location, history
- Approach zero energy waste
  - De-powering unused peripherals & power islands
- Heterogeneous cores & SP processor arrays (ULP)

0W vs 160mW (email, notifications)

10's mW vs 100's mW cores

Standby Power Saving

ideal 0%

22.5h @ 160mW = 3.6Wh   5Wh battery: 70%

default: email, skype   goal <10mW, 1kJ

Near-Zero-Power Standby Mode

- Zero-power always-on displays
  - Transflective, bistable
- Zero-power OS idling
  - Android, Meego, WP7 use 15mW
  - iPhone uses 5mW

TF/BS, TF/LCD, TF/OLED
5mW vs OLED 500mW   *100x*

better firmware 2mW vs 15mW
with hibernation 1mW vs 15mW   *15x*

Ultra-Low-Power Always-On Device

- Wearable accessory for notifications & voice
- Seamless quick switch over to companion
- Week-long battery life on small battery

4mW vs 200mW handset   *50x*

Energy Harvesting

Effectively Approach Unlimited Standby

➢ Casual Charging

- Wireless inductive chargers
- Instant charging for quick fix
- Solar for large surface areas
- Cross-device energy sharing

1W desk, nightstand, car — *5h full charge*

best case charge 3W for 1W tablet — *1h charge → 3h use*

➢ Anticipatory Preexecution

- Speculative pre-processing and pre-fetching
- Cloud based or cross-device based
- Context and user behavior model driven

40mW vs 1W — *25x*

Refactoring the Mobile Form Factors?

- **Tablet (avg 2W)**
  - Exploit large battery & connectivity
  - \- Runs offloaded apps from Phone
  - \- Gathers data for Wearable
  - \+ Display normally off
  - \+ 50 WH battery → 25 hours nonstop use
- **Phone (avg 200mW → avg 40mW)**
  - \+ Normally hibernating in standby
  - \+ Offload apps to Tablet
  - \+ Longer battery life: 5x battery life
  - \+ 5 WH battery → 125 hours nonstop use
- **Wearable (avg 3mW)**
  - \+ Always-on display
  - \+ Always fresh data feeds
  - \- Respond using Tablet or Phone
  - \+ Reduces avg power of Tablet & Phone
  - \+ 1.2 WH battery → 400 hours nonstop use

Multiple Devices, One Seamless Experience

18-600 Foundations of Computer Systems

Lecture 23: "Network Programming – Part 1"

John P. Shen & Zhiyi Yu (with Chris Inacio of SEI)
November 21, 2016

Next Time …

➢ Required Reading Assignment:
- **Chapter 11 of CS:APP (3rd edition) by Randy Bryant & Dave O'Hallaron.**

Electrical & Computer ENGINEERING