18-640 Foundations of Computer Architecture

Lecture 18:
"Performance and Power Iron Laws"

John Paul Shen
November 6, 2014

- Required Reading Assignment:
  - "Energy per Instruction Trends in Intel® Microprocessors," by Ed Grochowski, Murali Annavaram, 2006.
- Recommended Reading Assignments:
  - "Best of Both Latency and Throughput," by E. Grochowski, R. Ronen, J. Shen, H. Wang. In 22nd ICCD 2004.
  - "Mitigating Amdahl's Law through EPI Throttling," by M. Annavaram, E. Grochowski, J. Shen. In 32nd ISCA 2005.

Electrical & Computer ENGINEERING
Carnegie Mellon University

18-640 Foundations of Computer Architecture

Lecture 18:
"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

Electrical & Computer ENGINEERING
Carnegie Mellon University

Law #1 – Processor (Latency) Performance

- Time to execute a program: T (latency)

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

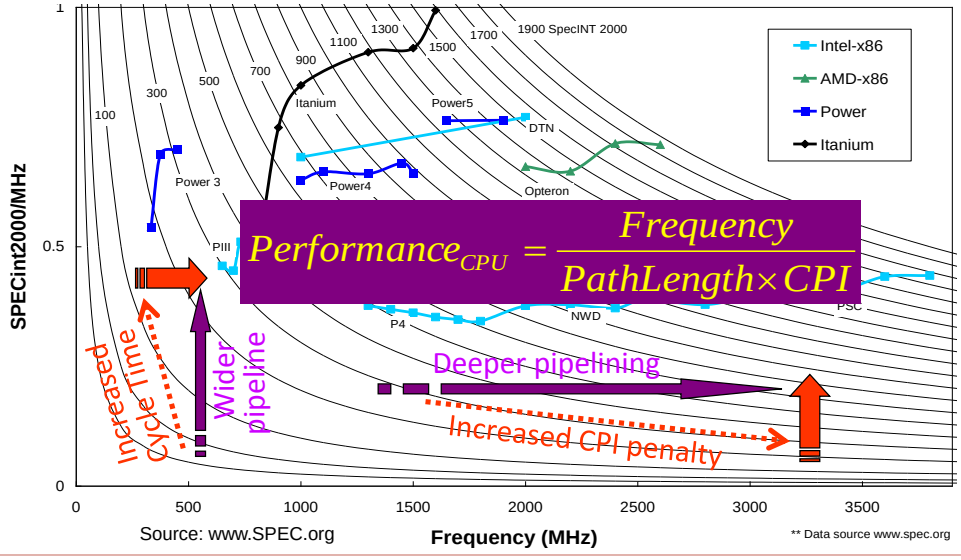
$$T = PathLength \times CPI \times CycleTime$$

- Processor performance: Perf = 1/T

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

[John DeVale & Bryan Black, 2005]

SPECint (Latency) Performance of Processors



Latency vs. Throughput Performance

- Reduce Latency of Application
  - Uni-processor, Single Program
  - Target Single-Thread Performance
  - Examples: SPEC, PC and Workstations
- Increase Throughput of System
  - Multi-processors, Many Threads/Tasks
  - Target Multi-threaded/Multi-tasking Throughput
  - Example: Database Transaction Processing

Law #2 – 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

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

Iron Law of Multiprocessor/Multicore Performance

- Multi-Core Performance:

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

- 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)

Law #3 – Multiprocessor Performance Scalability

- Multi-Core Speedup:

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

- A Rigorous Scalability Model:

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

Scalability Impedance Functions

- Multi-Core Speedup:

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

- Scalability Impedance Functions:

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

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

Real World Example: Database Performance Iron Law

- MP Database Performance: TPS

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

4-way SMP 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

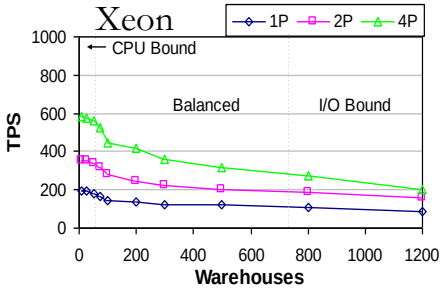
- Based on EMON Events
  - Separate User and OS components for each event
  - Use multiple long runs (20-min warm up, 10-min measurement)
  - Strive for standard deviation <5% for TPS & CPU utilization > 90%
  - Overall user execution time ~70-90%

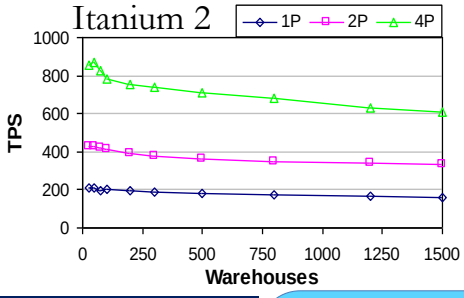
[Rich Hankins & Murali Annavaram, 2004]

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

TPS: Throughput Scaling

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



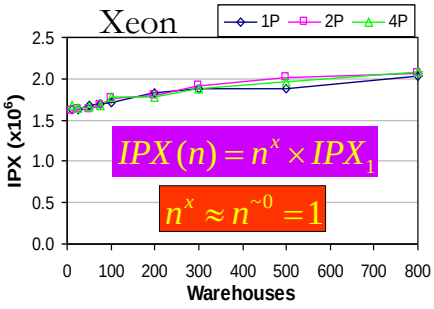


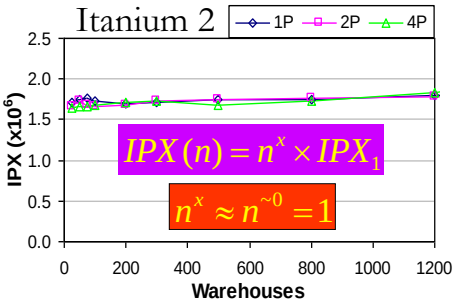
- TPS scales more linearly on Itanium2
  - Larger SGA implies slower I/O rate increase (Xeon I/O rate increases at 7KB/WH, & Itanium 2 at: 6KB/WH)
  - Bus utilization on Xeon higher than Itanium 2 (45% vs. 39%)
- Increasing I/O rate → more processes & context switches
  - Clients increase from 8 to 56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

IPX: Path-Length Scaling

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





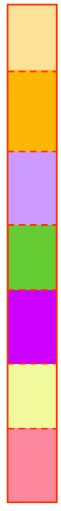
- Growth in IPX (quite linear) attributed to OS IPX increase
  - More I/O, more context switching
- User level IPX remains relatively constant in both systems
  - Code path through Oracle relatively constant
- Excluding NOPS, IPX at 25 WH similar for both systems!
- IPX growth less pronounced on Itanium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

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

Path-Length Contributions

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

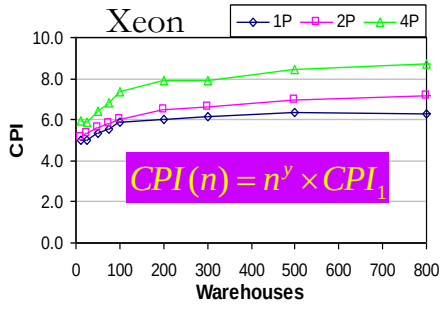
$$PL(n) \begin{cases} OS \begin{cases} I/O \\ Page\ Fault \\ Context\ Switch \end{cases} \\ APP \begin{cases} Original\ Code \\ Speculation \\ Replay \\ Synchronization \end{cases} \end{cases}$$

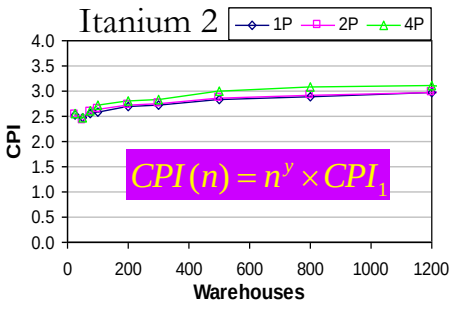


$$PL(n) = \left(n^x \times PL_1\right) \mid 1.0 \geq x \geq 0.0$$

CPI: Overhead Scaling

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



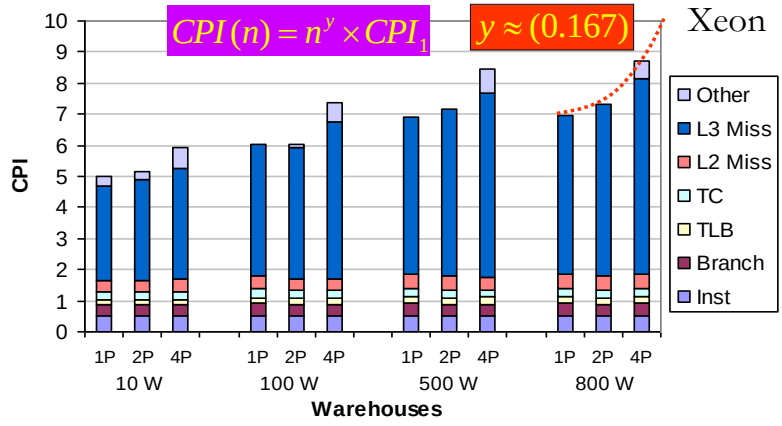


- CPI increases with a "knee" but less sharp on Itanium 2 !
- Overall CPI trend strongly determined by user CPI
  - User mode execution time more than 90% on IPF, 80% on Xeon
- Xeon CPI grows with P, Itanium 2 CPI does not
  - Growth attributed to higher bus utilization on Xeon

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

CPI Breakdown: Xeon

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



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

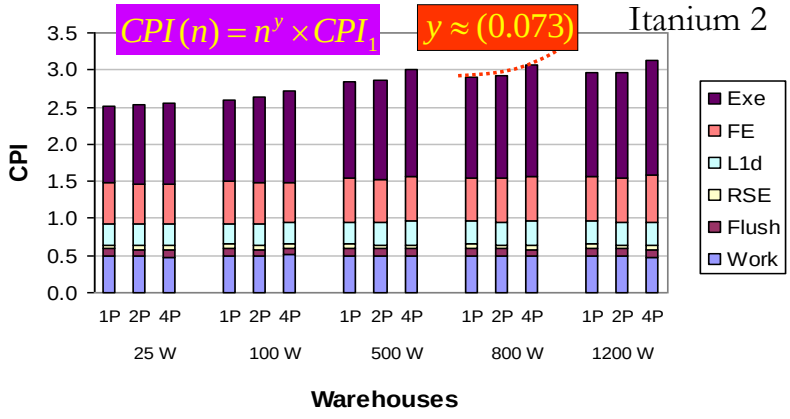
$$Speedup(n)_{MC} = \left(\frac{n}{n^x \times n^y}\right) = n^{0.833}$$

[Law #3]

CPI Breakdown: Itanium 2

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

$CPI(n) = n^{y} \times CPI_1$ $\quad y \approx (0.073)$



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

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

[Law #3]

CPI Contributions

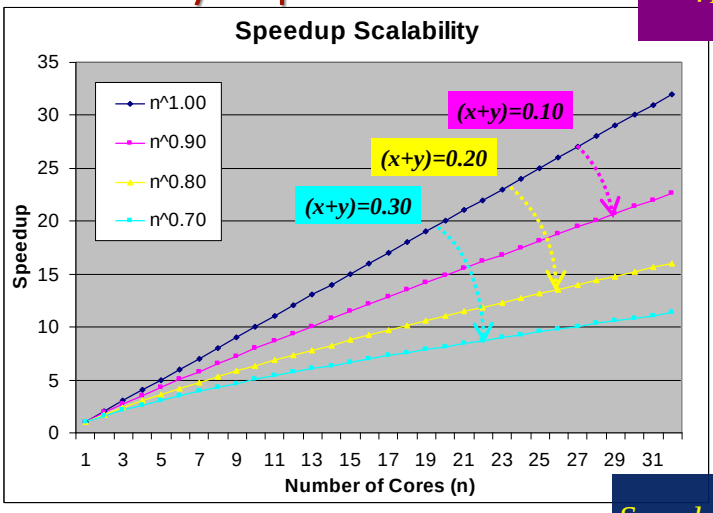
$$TPS = \frac{n \times Frequency}{IPX(n) \times CPI(n)}$$

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

$$CPI(n) = \left(n^{y} \times CPI_1\right) \mid 1.0 \geq y \geq 0.0$$

Scalability Impedance

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



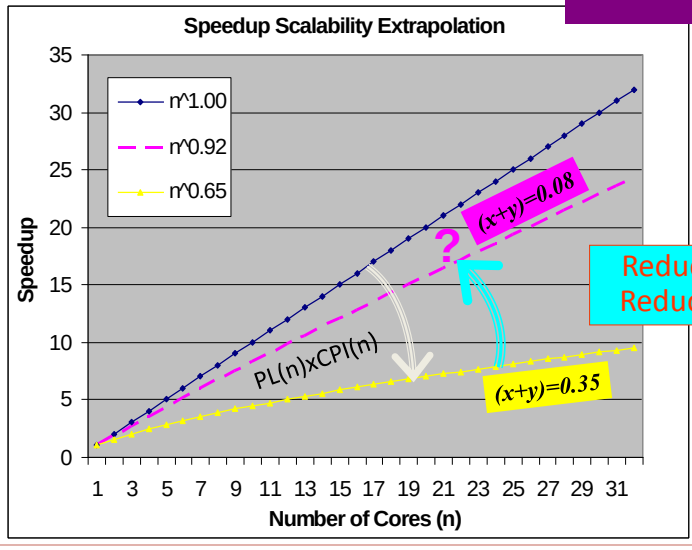
[Carole Dulong et al., 2005]

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

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

Scalability Headroom

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



Reduce PL(n)
Reduce CPI(n)

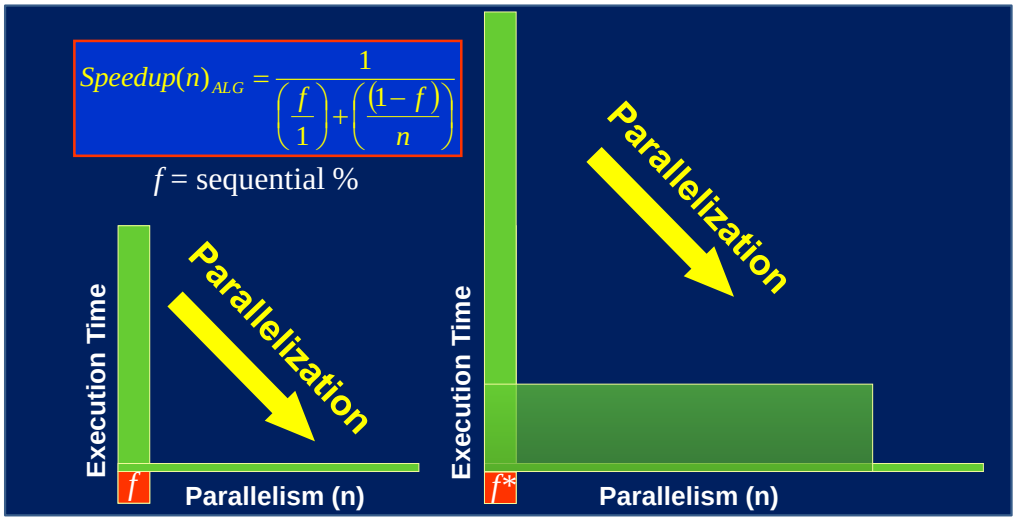
Conspiring Forces Against Multiprocessor Scaling Three Forms of Scalability Impedance

- Algorithm
  - Limitation of Language and Algorithm
  - Tyranny of Amdahl's Law (sequential bottleneck)
- Architecture (so far)
  - Increase of Path-Length Undermines Scalability
  - Increase of CPI 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)

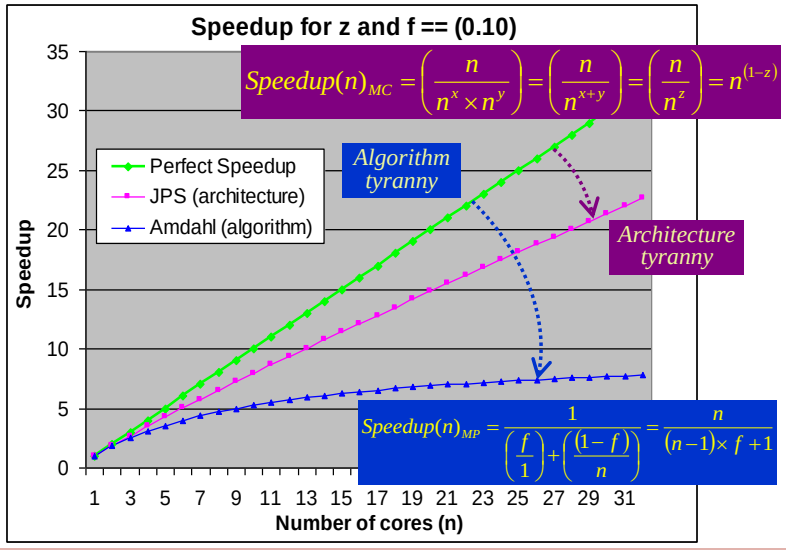
Algorithm Scalability: (Amdahl's Law)

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

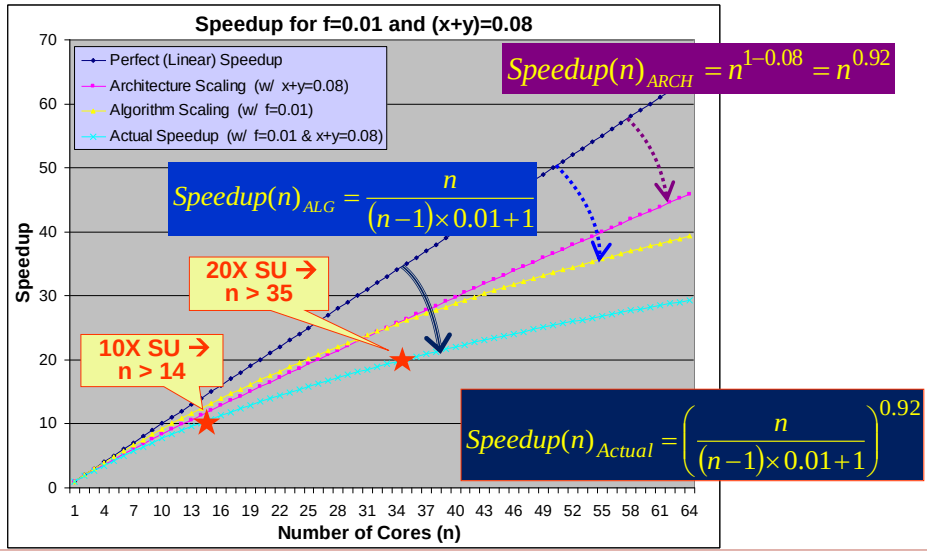
Architecture Scalability:

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

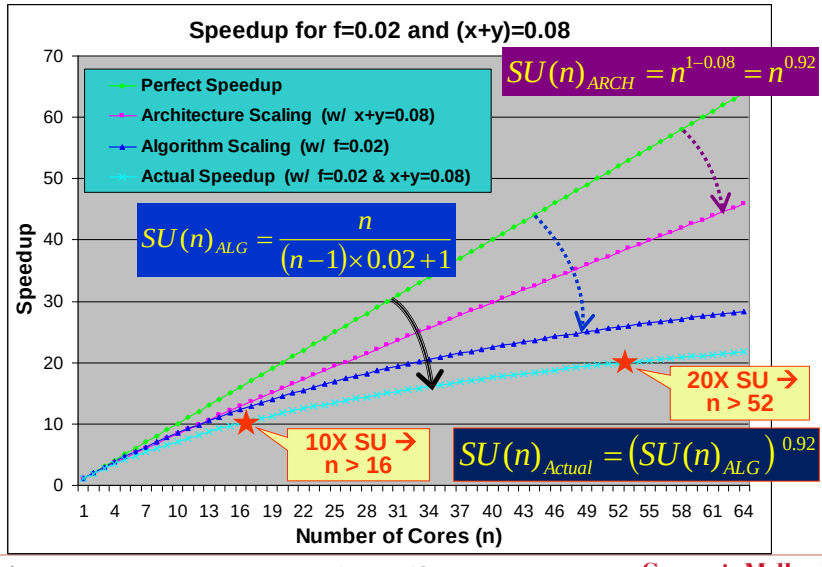
Relative (Additive) Degrees of Tyranny



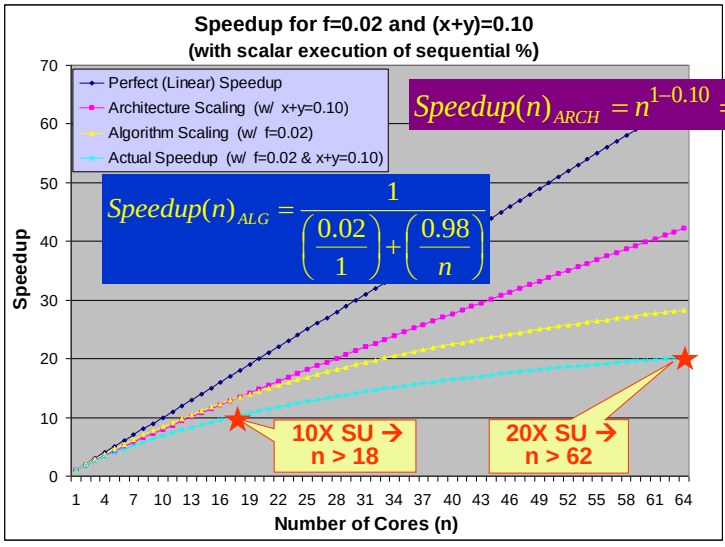
Combined Effect on Actual Speedup (f=0.01, x+y=0.08)



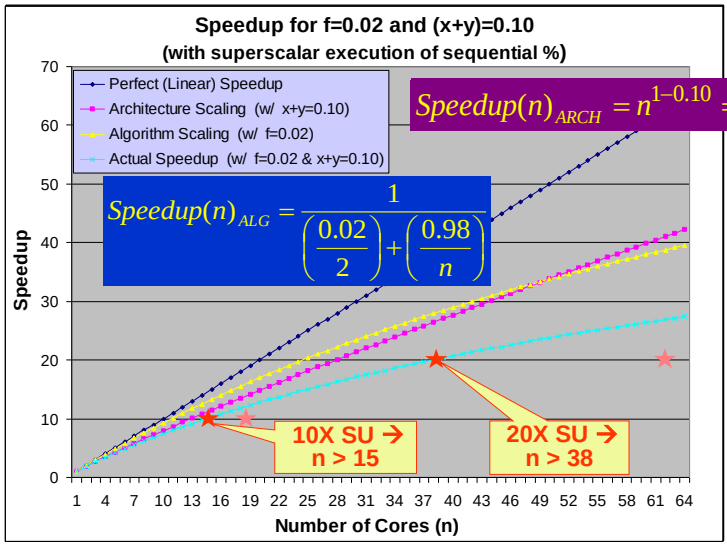
Combined Effect on Actual Speedup (f=0.02, x+y=0.08)



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



Impact of Latency Performance on MP Performance



Law #5 – Power and Performance

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

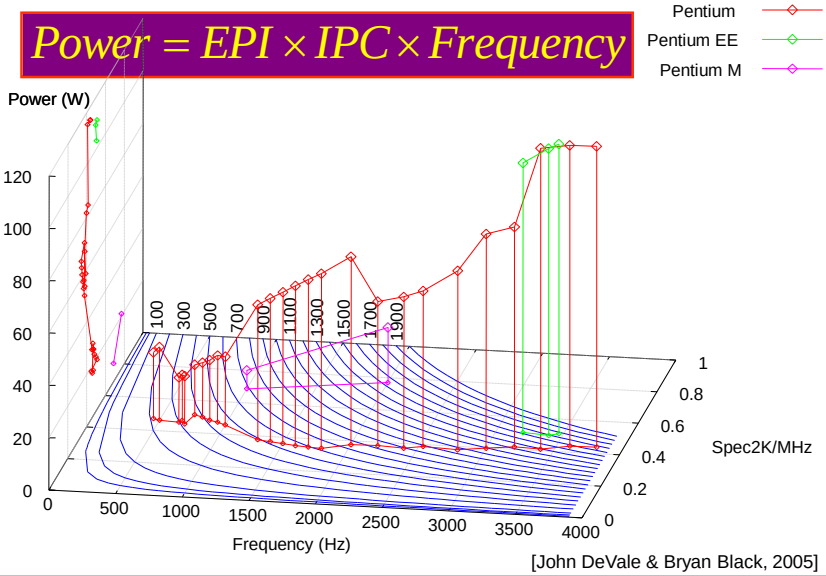
$$Power = EPI \times IPC \times Frequency$$

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

$$Power = EPI \times Performance \times PathLength$$

Iron Law of Power and Performance

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

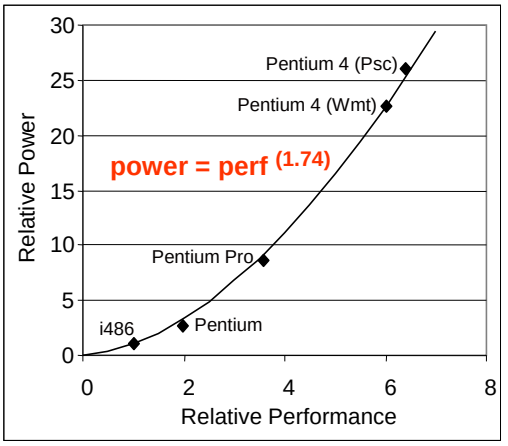


[John DeVale & Bryan Black, 2005]

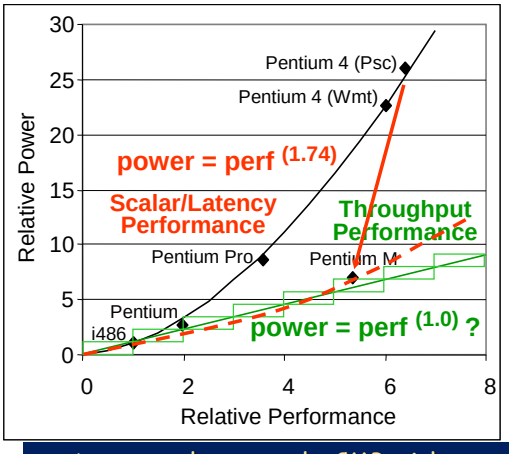
Power and Scalar (Latency) Performance

[Ed Grochowski, 2004]

- 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]

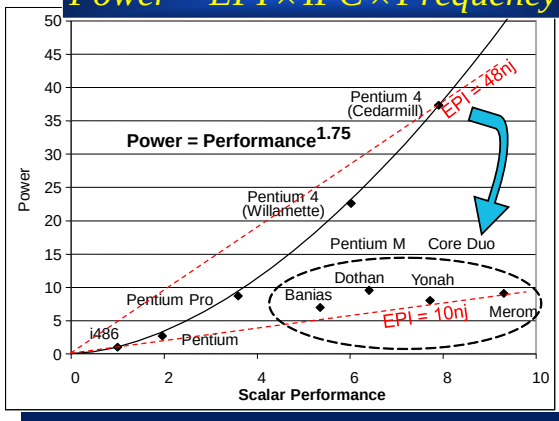
CPU	EPI
I486	7 nj
P5	10 nj
P6	17 nj
P4P-wmt	27 nj
P4P-psc	29 nj
Pentium M	9 nj

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

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



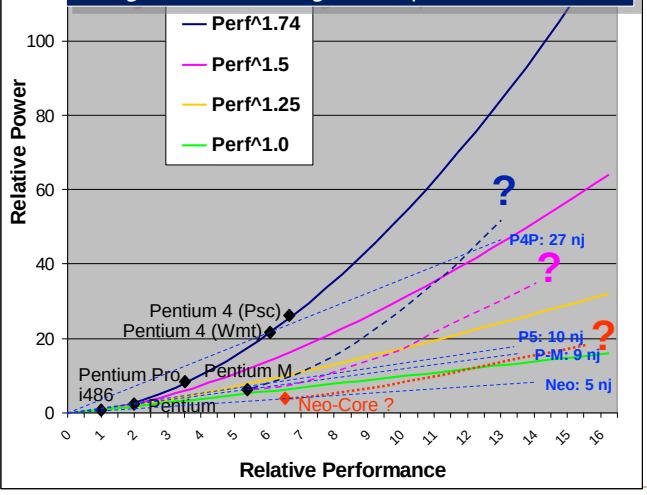
[Ed Grochowski, 2004]

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)

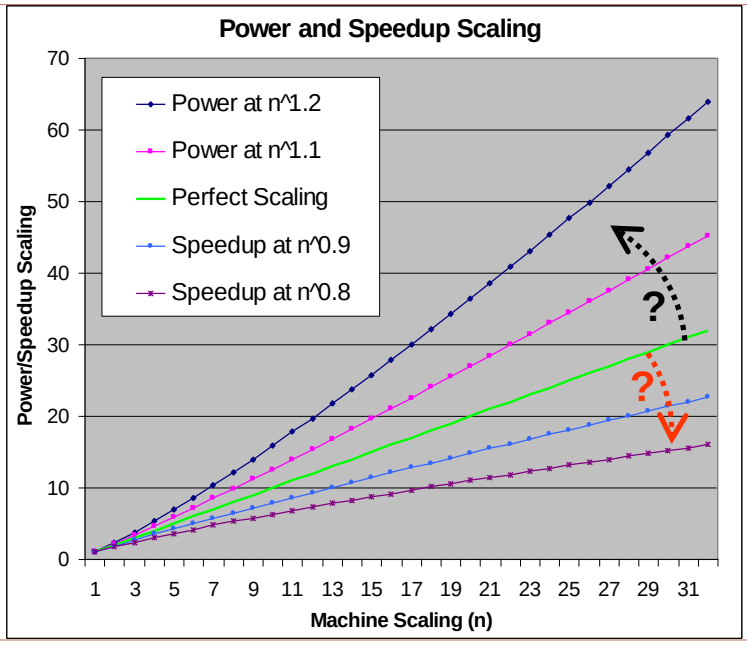
Power/Performance Scaling

- The issue is not small vs. big cores, nor in-order vs. out-of-order cores. The key metric is EPI.
- The ideal core: ultra-low EPI with best possible single-thread or single-core performance.



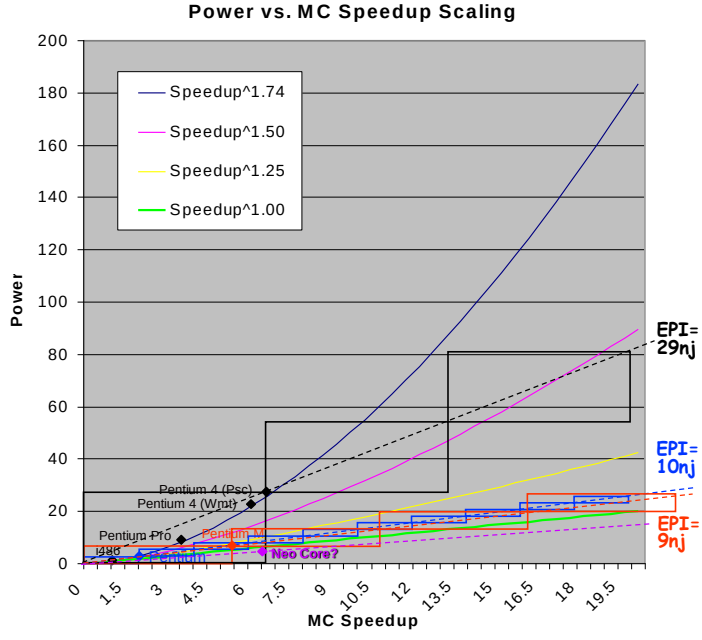
CPU	EPI	SU
i486	7 nj	1
P5	10 nj	2
P6	17 nj	3.5
P4P-wmt	27 nj	6
P4P-psc	29 nj	6.5
Pentium M	9 nj	5.5
Neo-Core	5 nj	6.5

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



Power scaling challenges:
- Low EPI cores
- Un-core scaling

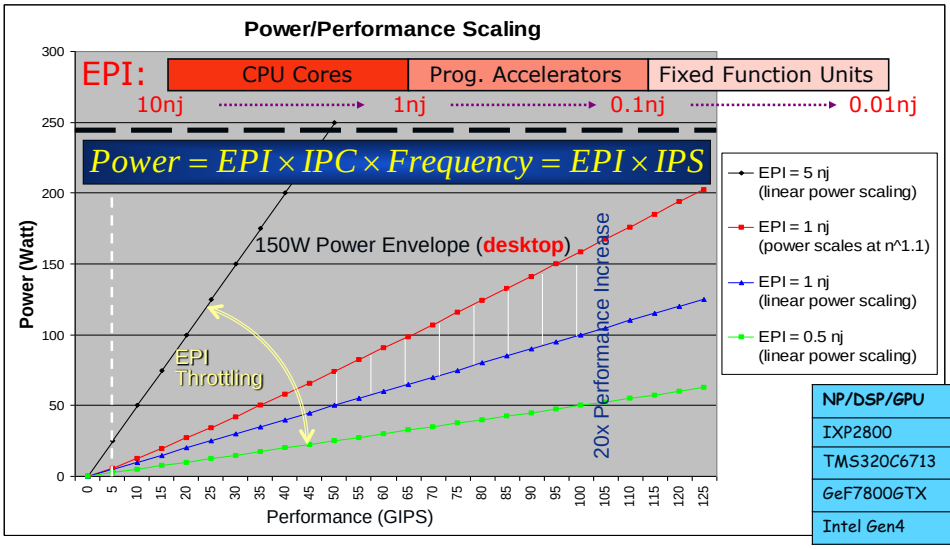
Speedup scaling challenges:
- Algorithm
  - Sequential %
- Architecture
  - PL scaling
  - CPI scaling



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

EPI Optimization for 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 likely THE scalability wall
- Research Challenges:
  - Sequential % mitigation for compelling workloads
  - Ultra-low EPI core with great ST performance
  - Un-core fabric with near-linear power scaling
- Un-core scaling is likely the new power goblin