ELEC 5200-001/6200-001
Computer Architecture and Design

Fall 2013

Performance of a Computer (Chapter 4)

Vishwani D. Agrawal & Victor P. Nelson
epartment of Electrical and Computer Engineering
Auburn University, Auburn, AL 36849

What is Performance?

- Response time: the time between the start and completion of a task.
- Throughput: the total amount of work done in a given time.
- Some performance measures:
  - MIPS (million instructions per second).
  - MFLOPS (million floating point operations per second), also GFLOPS, TFLOPS (10^{12}), etc.
  - SPEC (System Performance Evaluation Corporation) benchmarks.
  - LINPACK benchmarks, floating point computing, used for supercomputers.
  - Synthetic benchmarks.

Small and Large Numbers

Small			Large		
10^{-3}	milli	m	10^{3}	kilo	k
10^{-6}	micro	μ	10^{6}	mega	M
10^{-9}	nano	n	10^{9}	giga	G
10^{-12}	pico	p	10^{12}	tera	T
10^{-15}	femto	f	10^{15}	peta	P
10^{-18}	atto		10^{18}	exa	
10^{-21}	zepto		10^{21}	zetta	
10^{-24}	yocto		10^{24}	yotta	

Computer Memory Size

Number			bits	bytes
2^{10}	1,024	K	Kb	KB
2^{20}	1,048,576	M	Mb	MB
2^{30}	1,073,741,824	G	Gb	GB
2^{40}	1,099,511,627,776	T	Tb	TB

Units for Measuring Performance

- Time in seconds (s), microseconds (μs), nanoseconds (ns), or picoseconds (ps).
- Clock cycle
  - Period of the hardware clock
  - Example: one clock cycle means 1 nanosecond for a 1GHz clock frequency (or 1GHz clock rate)

  CPU time = (CPU clock cycles)/(clock rate)
- Cycles per instruction (CPI): average number of clock cycles used to execute a computer instruction.

Components of Performance

Components of Performance	Units
CPU time for a program	Time (seconds, etc.)
Instruction count	Instructions executed by the program
CPI	Average number of clock cycles per instruction
Clock cycle time	Time period of clock (seconds, etc.)

Time, While You Wait, or Pay For

- *CPU time* is the time taken by CPU to execute the program. It has two components:
  – *User CPU time* is the time to execute the instructions of the program.
  – *System CPU time* is the time used by the operating system to run the program.
- *Elapsed time (wall clock time)* is the time between the start and end of a program.

Example: Unix "time" Command

90.7u	12.9s	2:39	65%
User CPU time in seconds	System CPU time in seconds	Elapsed time In min:sec	CPU time as percent of elapsed time

$$\frac{90.7 + 12.9}{159} \times 100 = 65\%$$

Computing CPU Time

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$= \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{1 second}}{\text{Clock rate}}$$

Comparing Computers C1 and C2

- Run the same program on C1 and C2. Suppose both computers execute the same number (N) of instructions:
  - C1: CPI = 2.0, clock cycle time = 1 ns

    $$\text{CPU time(C1)} = N \times 2.0 \times 1 = 2.0N \text{ ns}$$

  - C2: CPI = 1.2, clock cycle time = 2 ns

    $$\text{CPU time(C2)} = N \times 1.2 \times 2 = 2.4N \text{ ns}$$

- CPU time(C2)/CPU time(C1) = $2.4N/2.0N = 1.2$, therefore, C1 *is* 1.2 *times faster than* C2.
- Result can vary with the choice of program.

Comparing Program Codes I & II

- Code size for a program:
  - **Code I has 5 million instructions**
  - **Code II has 6 million instructions**
  - **Code I is more efficient. *Is it?***
- Suppose a computer has three types of instructions: A, B and C.
- CPU cycles (code I) = 10 million
- CPU cycles (code II) = 9 million
- *Code II is more efficient.*
  - CPI(I) = 10/5 = 2
  - CPI(II) = 9/6 = 1.5
  - *Code II is more efficient.*
- *Caution:* Code size is a misleading indicator of performance.

Instr. Type	CPI
A	1
B	2
C	3

Code	Instruction count in million			
	Type A	Type B	Type C	Total
I	2	1	2	5
II	4	1	1	6

Rating of a Computer

- MIPS: million instructions per second

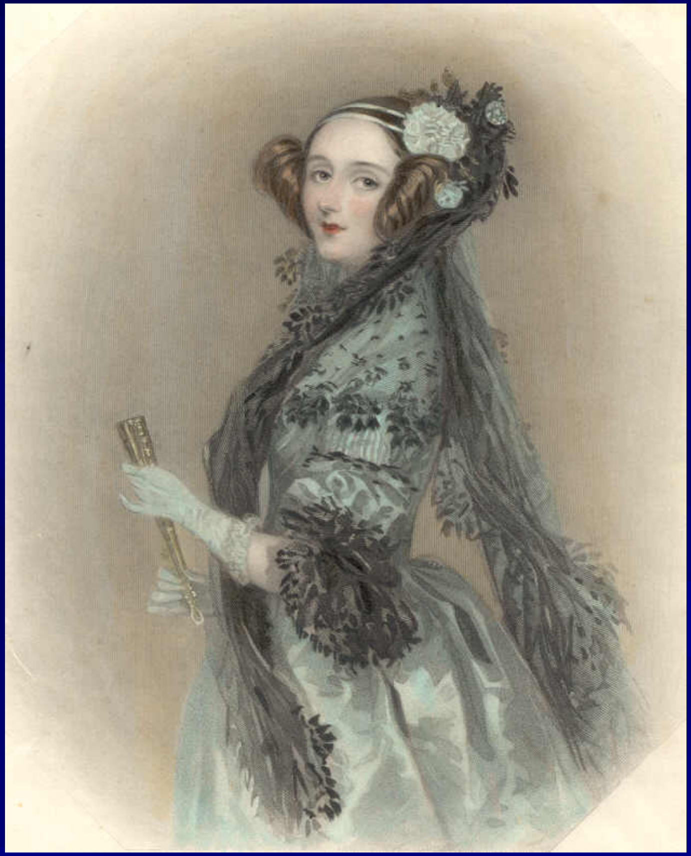
$$\text{MIPS} = \frac{\text{Instruction count of a program}}{\text{Execution time} \times 10^6}$$

- MIPS rating of a computer is relative to a program.
- Standard programs for performance rating:
  - Synthetic benchmarks
  - SPEC benchmarks (System Performance Evaluation Corporation)

Synthetic Benchmark Programs

- Artificial programs that emulate a large set of typical "real" programs.
- Whetstone benchmark – Algol and Fortran.
- Dhrystone benchmark – Ada and C.
- Disadvantages:
  - No clear agreement on what a typical instruction mix should be.
  - Benchmarks do not produce meaningful result.
  - Purpose of rating is defeated when compilers are written to optimize the performance rating.

Ada



Lady Augusta Ada Byron, *Countess of Lovelace* (1815-1852), daughter of Lord Byron (the poet who spent some time in a Swiss jail – in Chillon, not too far from Lausanne...). She was the assistant and patron of Charles Babbage; she wrote programs for his "Analytical Engine."

An original print from its time.
http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/pictures.html

Misleading Compilers

- Consider a computer with a clock rate of 1 GHz.
- Two compilers produce the following instruction mixes for a program:

Code from	Instruction count (billions)				CPU clock cycles	CPI	CPU time* (seconds)	MIPS**
	Type A	Type B	Type C	Total				
Compiler 1	5	1	1	7	10×10^9	1.43	10	700
Compiler 2	10	1	1	12	15×10^9	1.25	15	800

Instruction types – A: 1-cycle, B: 2-cycle, C: 3-cycle

* CPU time = CPU clock cycles/clock rate

** MIPS = (Total instruction count/CPU time) $\times 10^{-6}$

Peak and Relative MIPS Ratings

- Peak MIPS
  - Choose an instruction mix to minimize CPI
  - The rating can be too high and unrealistic for general programs
- Relative MIPS: Use a reference computer system

$$\text{Relative MIPS} = \frac{\text{Time(ref)}}{\text{Time}} \times \text{MIPS(ref)}$$

Historically, VAX-11/ 780, believed to have a 1 MIPS performance, was used as reference.

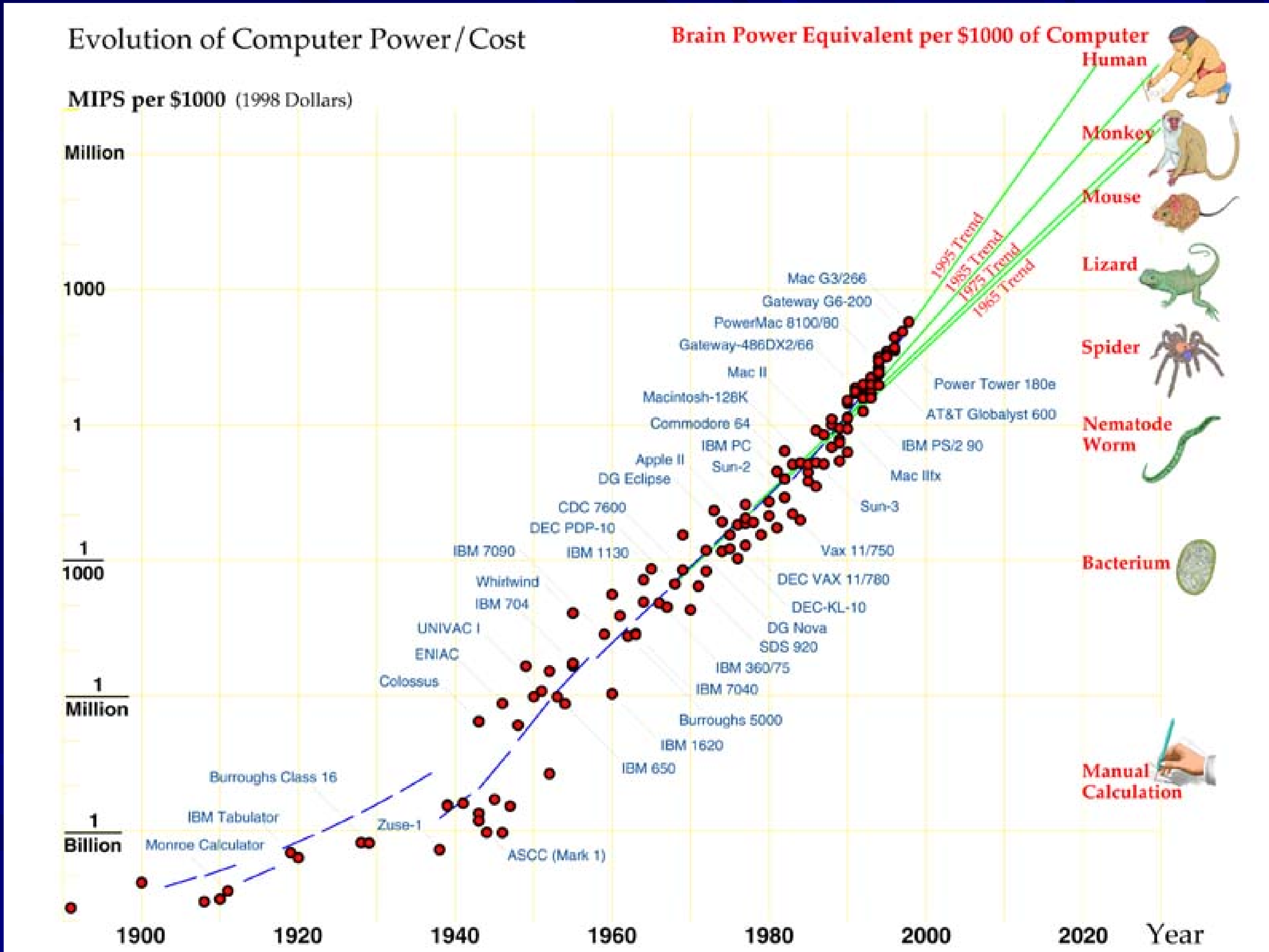
Wĕbopēdia on *MIPS*

- Acronym for ***m**illion **i**nstructions **p**er **s**econd*. An old measure of a computer's speed and power, MIPS measures roughly the number of machine instructions that a computer can execute in one second.
- In fact, some people jokingly claim that MIPS really stands for ***M**eaningless **I**ndicator of **P**erformance.*
- Despite these problems, a MIPS rating can give you a general idea of a computer's speed. The IBM PC/XT computer, for example, is rated at ¼ MIPS, while Pentium-based PCs run at over 100 MIPS.

A 1994 MIPS Rating Chart

Computer	MIPS	Price	$/MIPS
1975 IBM mainframe	10	$10M	1M
1976 Cray-1	160	$20M	125K
1979 DEC VAX	1	$200K	200K
1981 IBM PC	0.25	$3K	12K
1984 Sun 2	1	$10K	10K
1994 Pentium PC	66	$3K	46
1995 Sony PCX video game	500	$500	1
1995 Microunity set-top	1,000	$500	0.5

New York Times, April 20, 1994

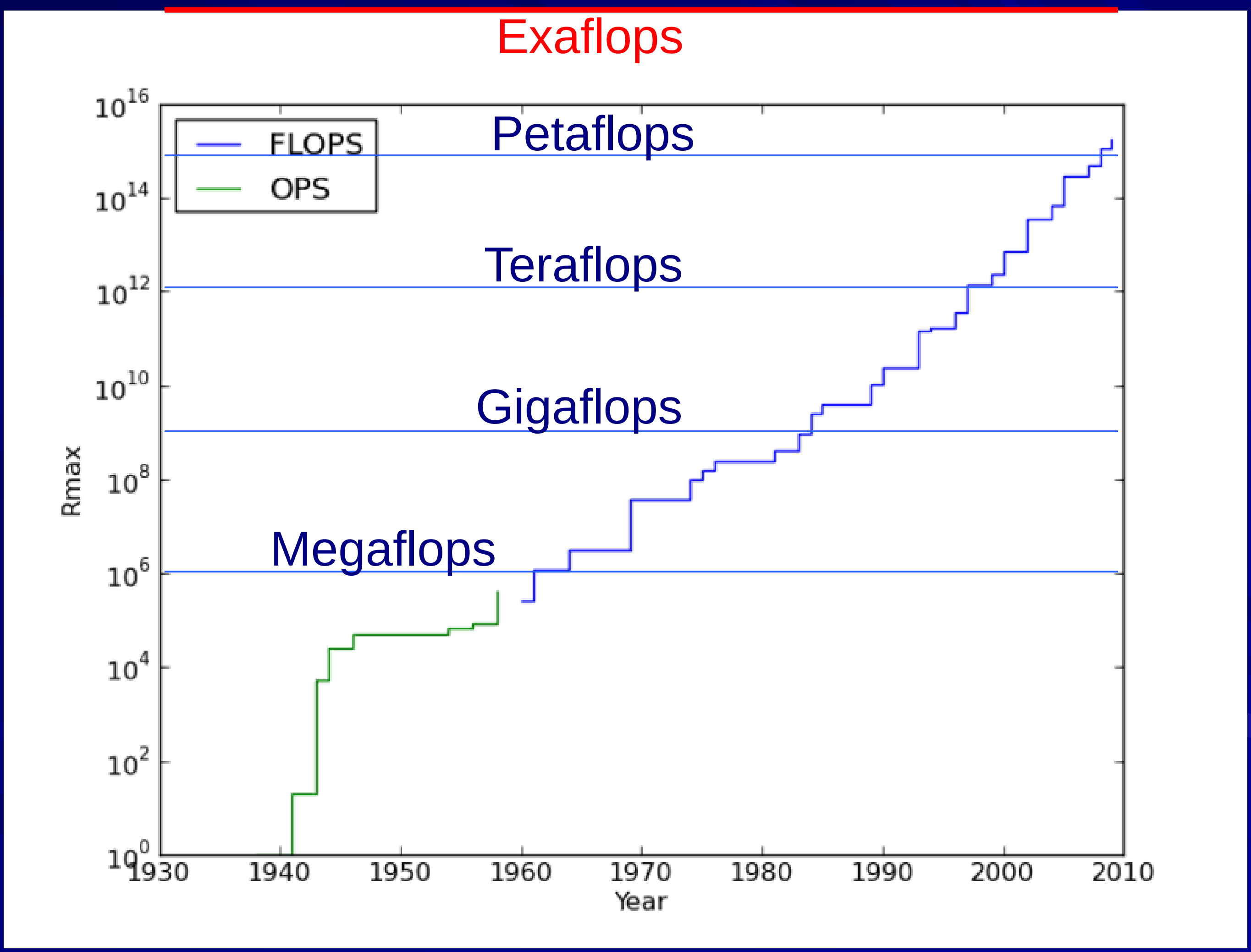

Evolution of Computer Power / Cost
Brain Power Equivalent per $1000 of Computer
MIPS per $1000 (1998 Dollars)
Million
1000
1
1/1000
1/Million
1/Billion
1900
1920
1940
1960
1980
2000
2020
Year
Human
Monkey
Mouse
Lizard
Spider
Nematode Worm
Bacterium
Manual Calculation
1995 Trend
1985 Trend
1975 Trend
1965 Trend
Mac G3/266
Gateway G6-200
PowerMac 8100/80
Gateway-486DX2/66
Mac II
Macintosh-128K
Commodore 64
IBM PC
Sun-2
Apple II
DG Eclipse
CDC 7600
DEC PDP-10
IBM 7090
IBM 1130
Whirlwind
IBM 704
UNIVAC I
ENIAC
Colossus
Burroughs Class 16
IBM Tabulator
Monroe Calculator
Zuse-1
ASCC (Mark 1)
Power Tower 180e
AT&T Globalyst 600
IBM PS/2 90
Mac IIfx
Sun-3
Vax 11/750
DEC VAX 11/780
DEC-KL-10
DG Nova
SDS 920
IBM 360/75
IBM 7040
Burroughs 5000
IBM 1620
IBM 650

MFLOPS (megaFLOPS)

$$\text{MFLOPS} = \frac{\text{Number of floating-point operations in a program}}{\text{Execution time} \times 10^6}$$

- Only floating point operations are counted:
  - Float, real, double; add, subtract, multiply, divide
- MFLOPS rating is relevant in scientific computing. For example, programs like a compiler will measure almost 0 MFLOPS.
- Sometimes misleading due to different implementations. For example, a computer that does not have a floating-point divide, will register many FLOPS for a division.

Supercomputer Performance



http://en.wikipedia.org/wiki/Supercomputer

Top Supercomputers, June 2012

www.top500.org

Rank	Name	Location	Cores	Clock GHz	Max. Pflops	Power MW	Eff. Pflops/ MJoule
1	Titan/Cray	Oak Ridge	560,640	2.2	27.11	8.21	3.30
2	Sequoia	IBM USA	1,572864	1.6	16.30	7.89	2.07
3	K computer	Fujitsu Japan	795,024	2.0	10.50	12.66	0.83
4	Mira	IBM USA	786,432	1.6	8.16	3.95	2.07
5	SuperMUC	IBM Germany	147,456	2.7	2.90	3.52	0.82

N. Leavitt, “Big Iron Moves Toward Exascale Computing,” *Computer*, vol. 45, no. 11, pp. 14-17, Nov. 2012.

The Future

Erik P. DeBenedictis of Sandia National Laboratories theorizes that a zettaflops (10^{21}) (one sextillion FLOPS) computer is required to accomplish full weather modeling, which could cover a two week time span accurately. Such systems might be built around 2030.

http://en.wikipedia.org/wiki/Supercomputer

Performance

- Performance is measured for a given program or a set of programs:

$$\text{Av. } \textit{execution time} = (1/n) \sum_{i=1}^{n} \textit{Execution time (program } i\textit{)}$$

or

$$\textit{Av. execution time} = \left[\prod_{i=1}^{n} \textit{Execution time (program } i\textit{)} \right]^{1/n}$$

- Performance is inverse of execution time:

$$\textit{Performance} = 1/(\textit{Execution time})$$

Geometric vs. Arithmetic Mean

- Reference computer times of n programs: r1, . . . , rn
- Times of n programs on the computer under evaluation: T1, . . . , Tn
- Normalized times: T1/r1, . . . , Tn/rn
- Geometric mean $= \{(T1/r1) \ldots (Tn/rn)\}^{1/n}$

  $$= \frac{\{T1 \ldots Tn\}^{1/n}}{\{r1 \ldots rn\}^{1/n}}$$ Used

- Arithmetic mean $= \{(T1/r1)+ \ldots +(Tn/rn)\}/n$

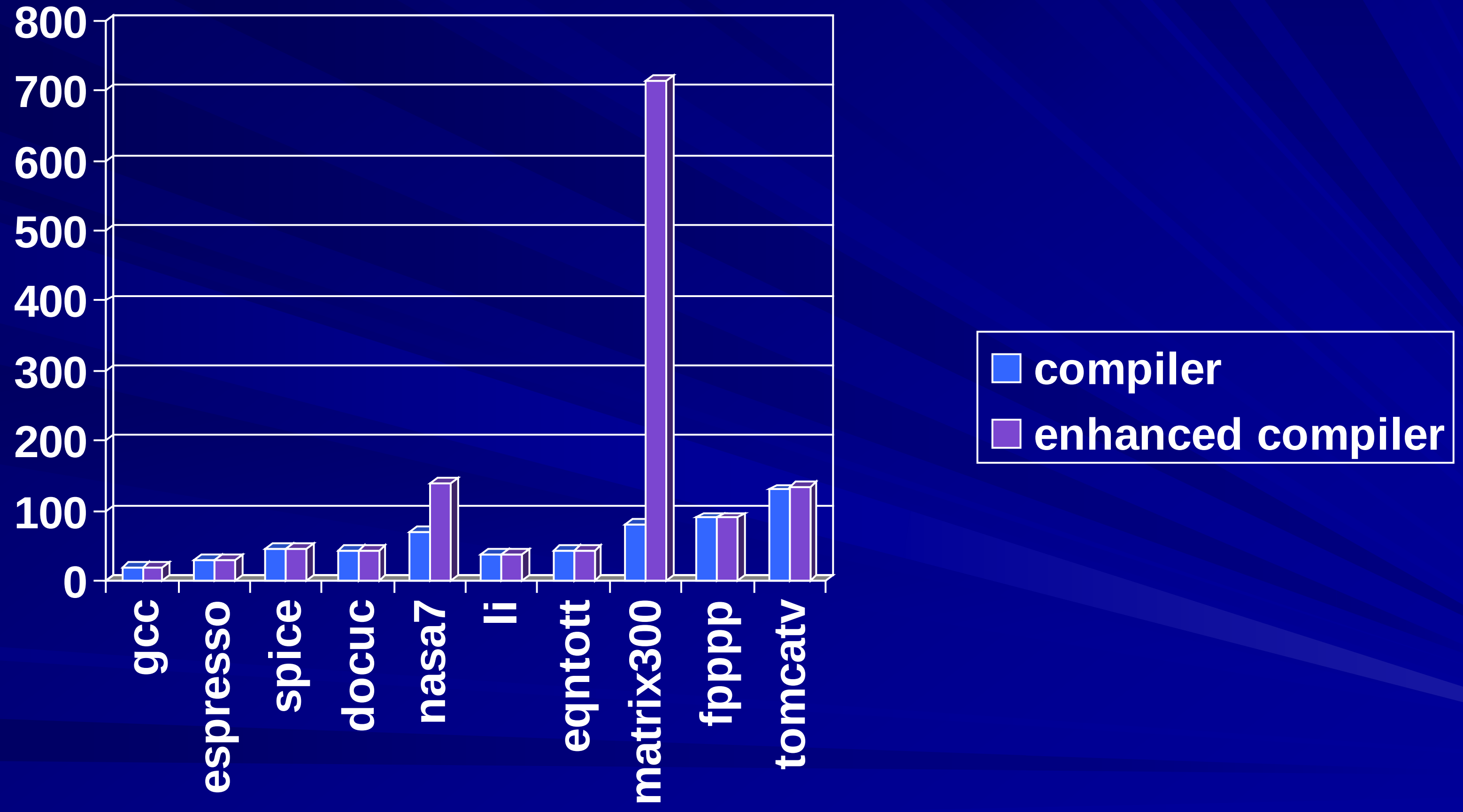
  $$\neq \frac{\{T1+ \ldots +Tn\}/n}{\{r1+ \ldots +rn\}/n}$$ Not used

J. E. Smith, "Characterizing Computer Performance with a Single Number," *Comm. ACM*, vol. 31, no. 10, pp. 1202-1206, Oct. 1988.

SPEC Benchmarks

- System Performance Evaluation Corporation (SPEC)
- SPEC89
  - 10 programs
  - SPEC performance ratio relative to VAX-11/780
  - One program, matrix300, dropped because compilers could be engineered to improve its performance.
  - *www.spec.org*

SPEC89 Performance Ratio for IBM Powerstation 550



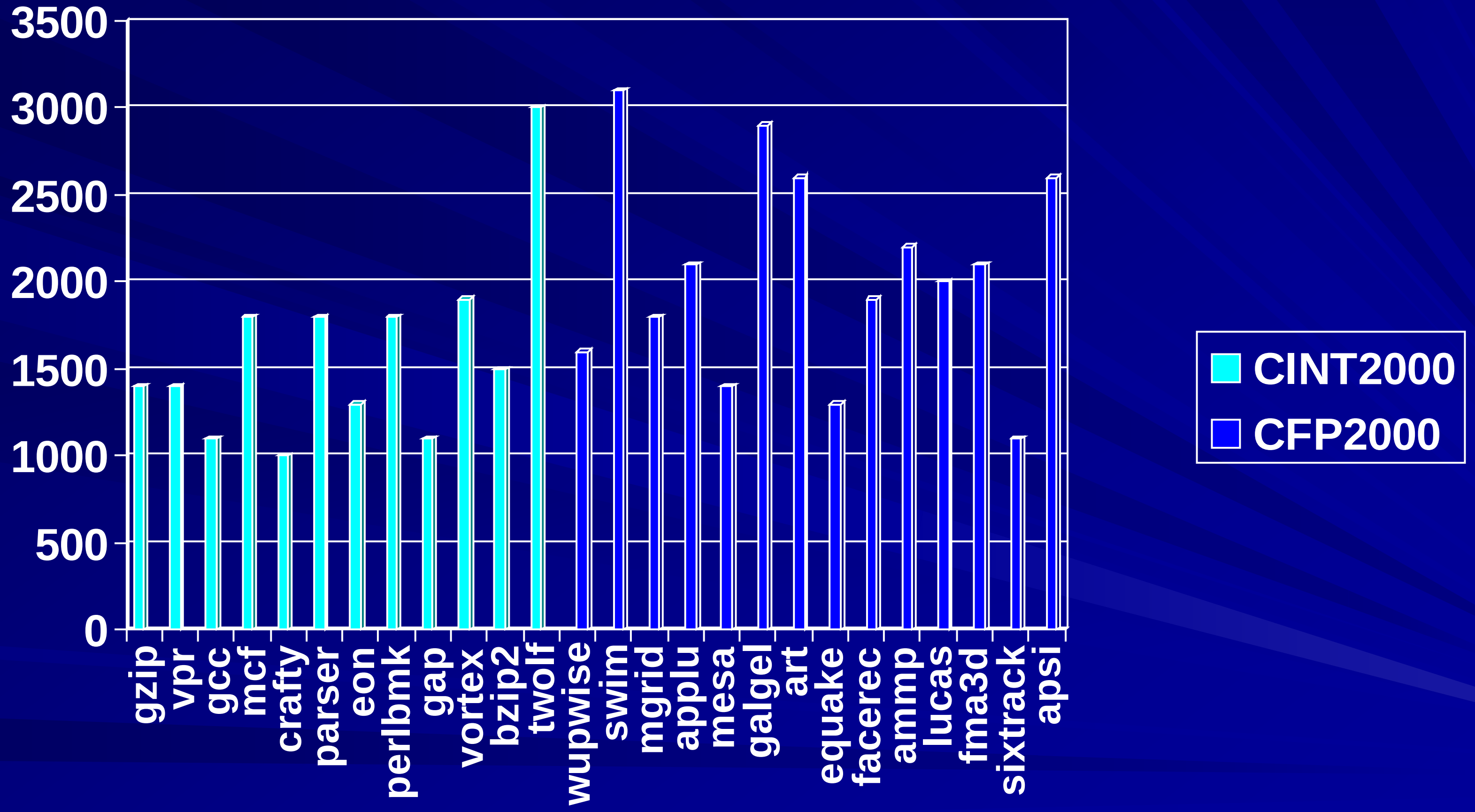
SPEC95 Benchmarks

- Eight integer and ten floating point programs, *SPECint95* and *SPECfp95*.
- Each program run time is normalized with respect to the run time of *Sun SPARCstation 10/40* – the ratio is called *SPEC ratio*.
- *SPECint95* and *SPECfp95* summary measurements are the geometric means of SPEC ratios.

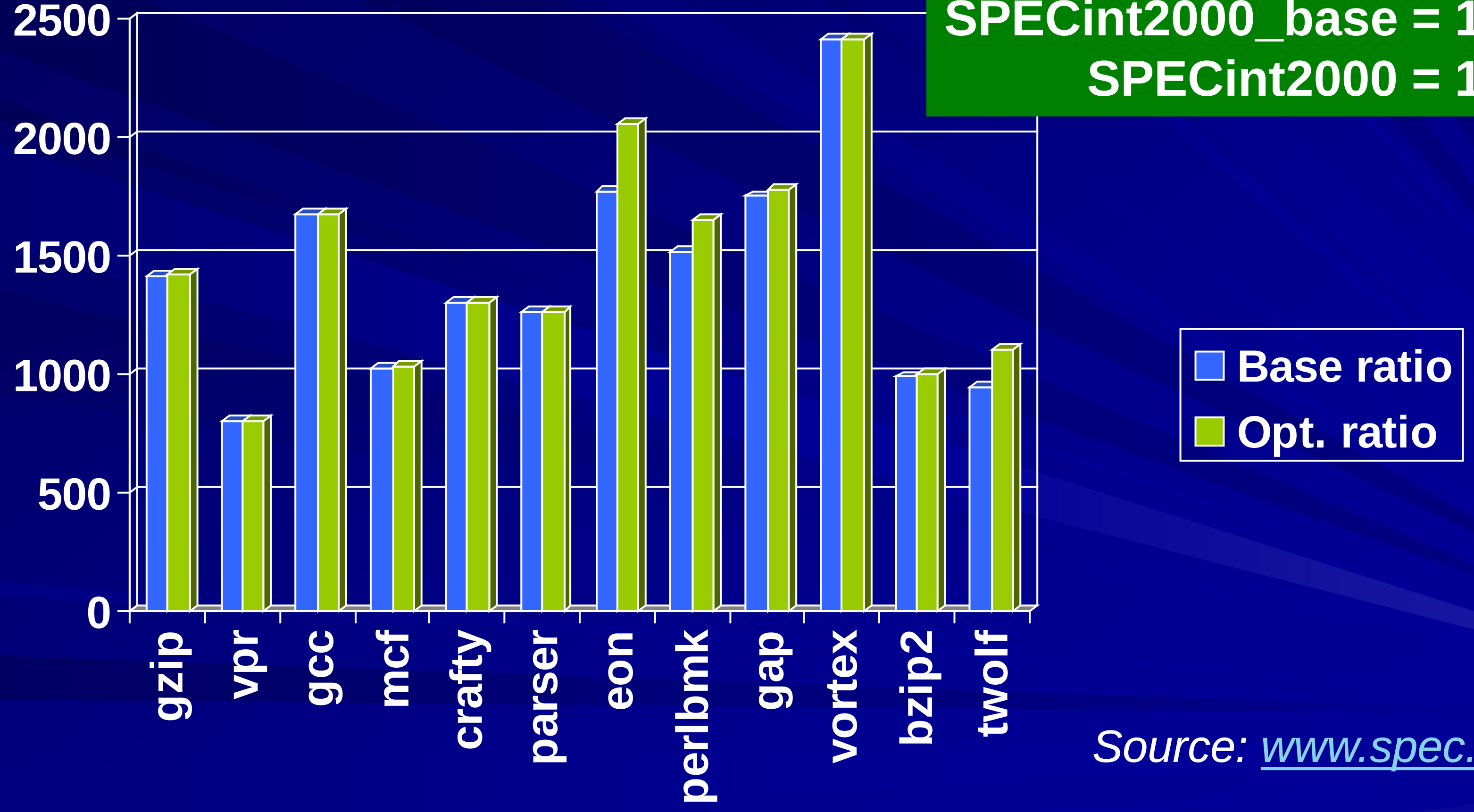
SPEC CPU2000 Benchmarks

- Twelve integer and 14 floating point programs, *CINT2000* and *CFP2000*.
- Each program run time is normalized to obtain a *SPEC ratio* with respect to the run time on *Sun Ultra 5_10 with a 300MHz processor*.
- *CINT2000* and *CFP2000* summary measurements are the geometric means of SPEC ratios.

Reference CPU: Sun Ultra 5_10 300MHz Processor



CINT2000: 3.4GHz Pentium 4, HT Technology (D850MD Motherboard)



SPECint2000_base = 1341
SPECint2000 = 1389

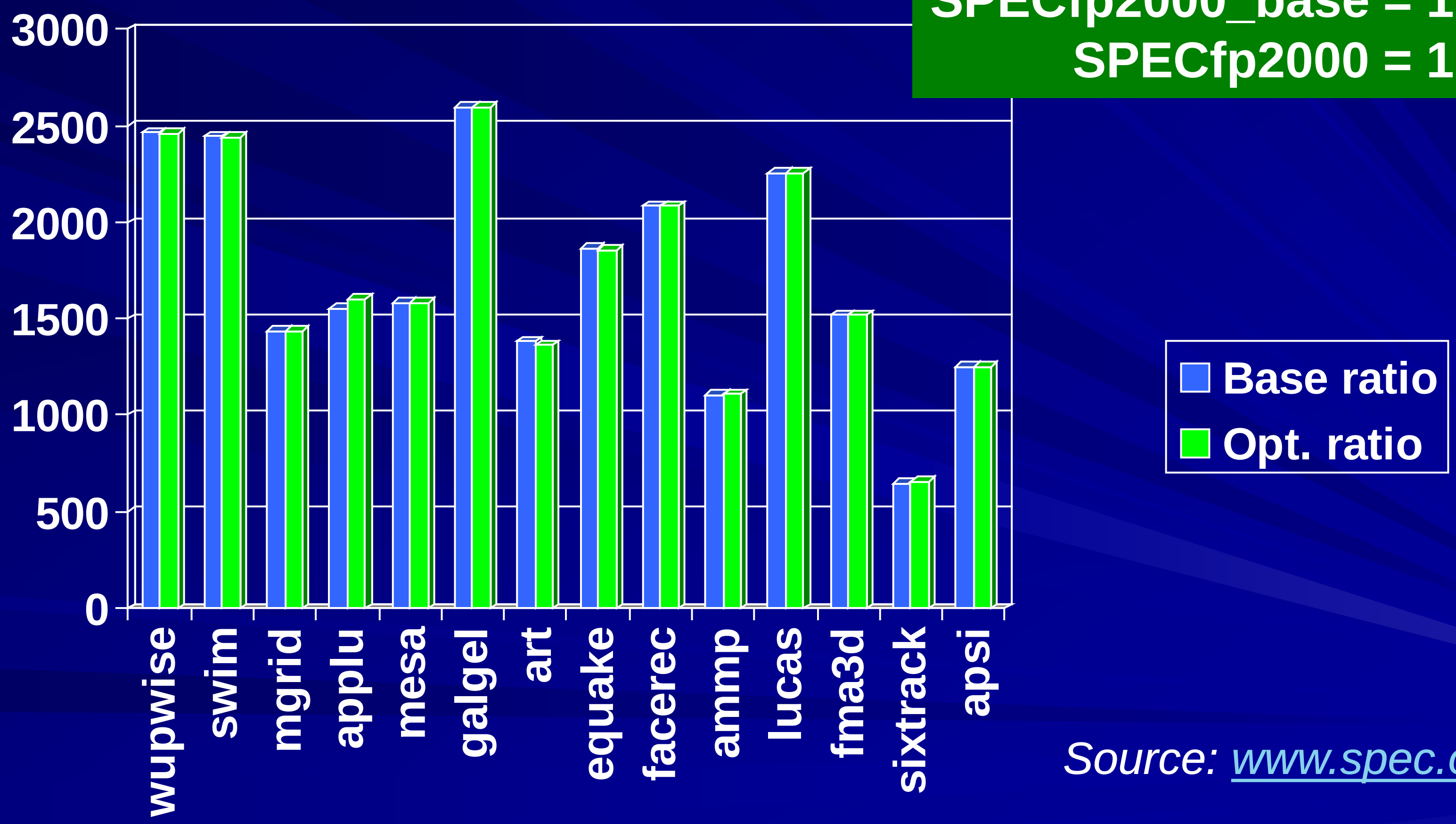
Source: www.spec.org

Two Benchmark Results

- Baseline: A uniform configuration not optimized for specific program:
  - Same compiler with same settings and flags used for all benchmarks
  - Other restrictions
- Peak: Run is optimized for obtaining the peak performance for each benchmark program.

CFP2000: 3.6GHz Pentium 4, HT Technology (D925XCV/AA-400 Motherboard)

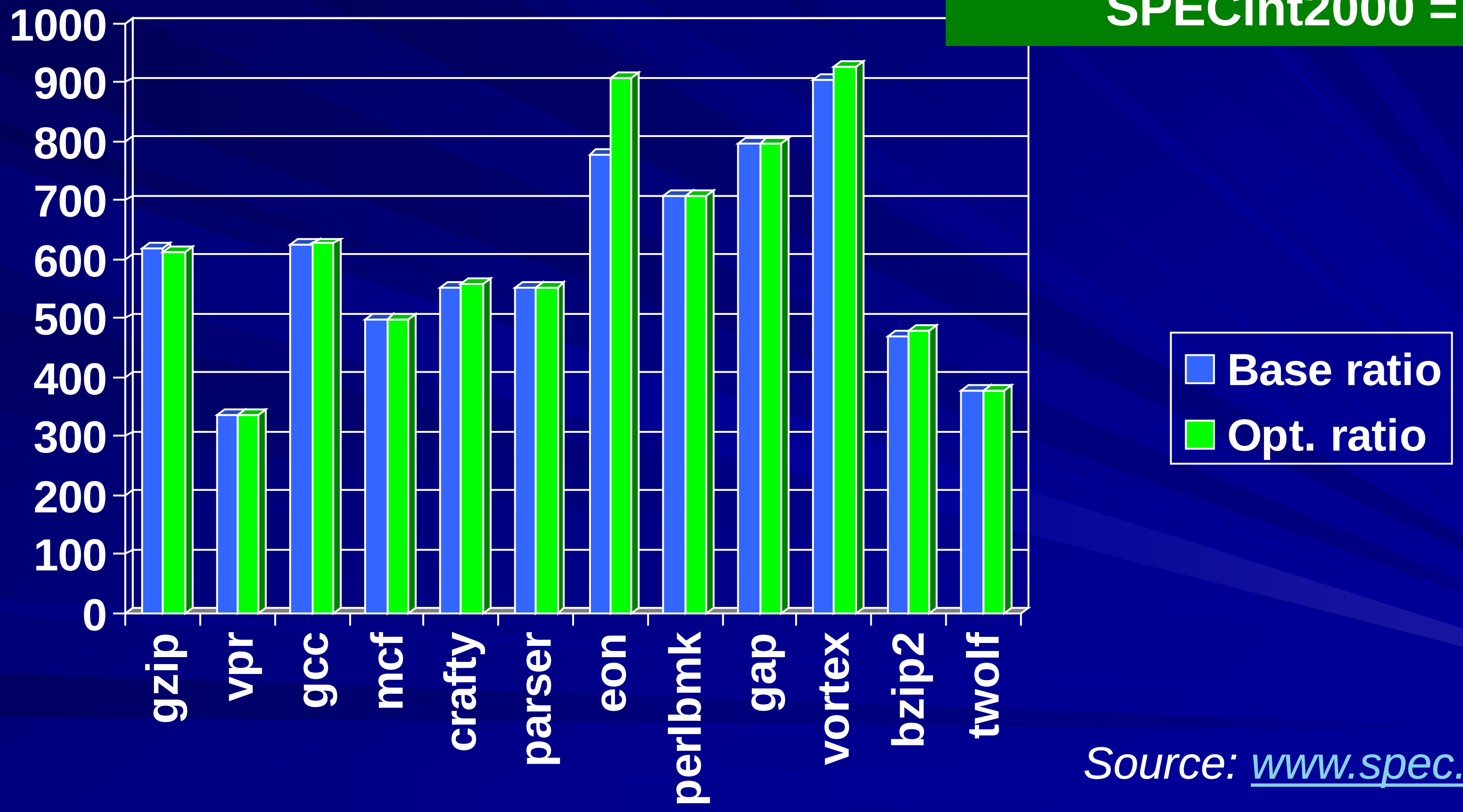
SPECfp2000_base = 1627
SPECfp2000 = 1630



Source: www.spec.org

CINT2000: 1.7GHz Pentium 4 (D850MD Motherboard)

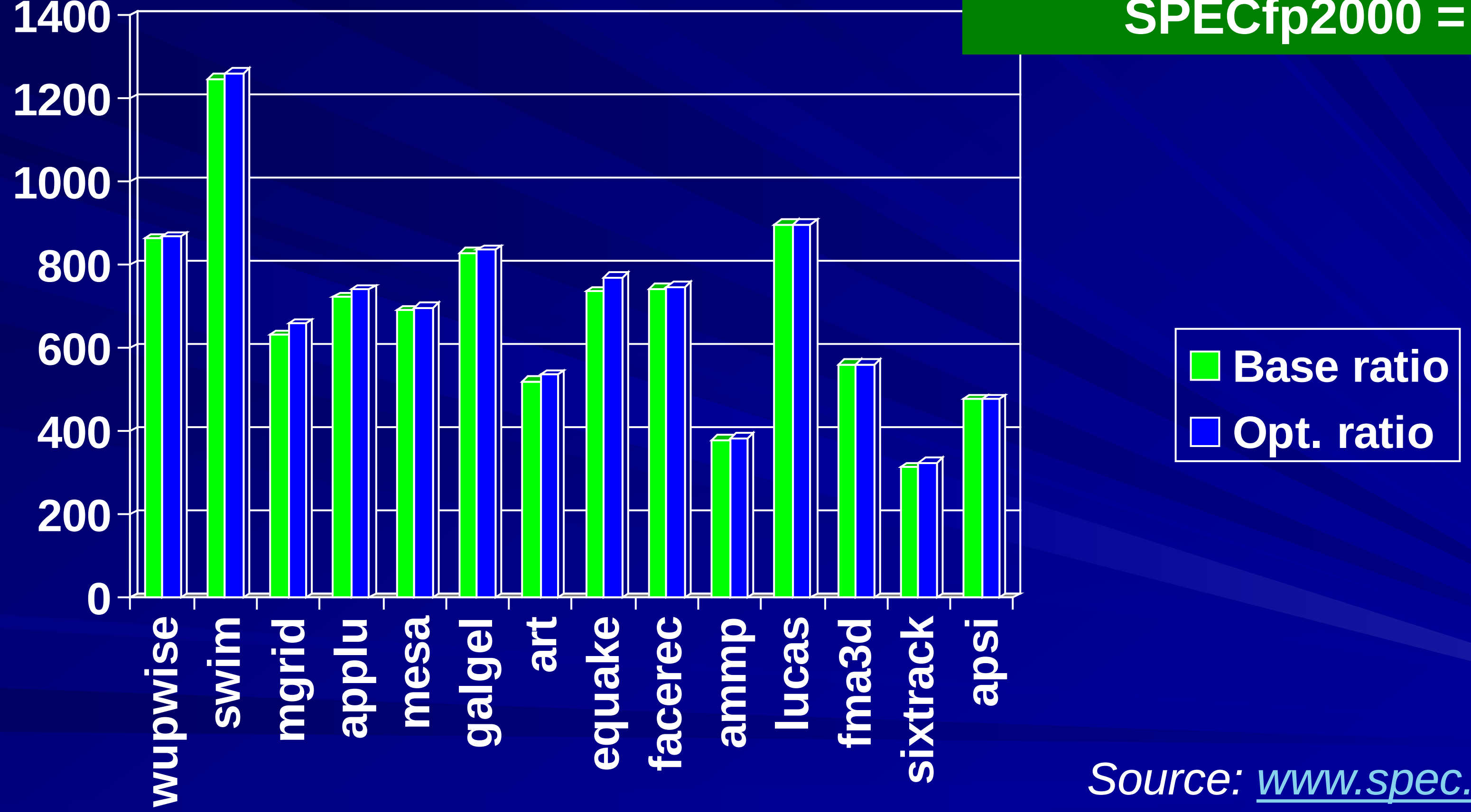
SPECint2000_base = 579
SPECint2000 = 588



Source: www.spec.org

CFP2000: 1.7GHz Pentium 4 (D850MD Motherboard)

SPECfp2000_base = 648
SPECfp2000 = 659



Source: www.spec.org

Additional SPEC Benchmarks

- SPECweb99: measures the performance of a computer in a networked environment.
- Energy efficiency mode: Besides the execution time, energy efficiency of SPEC benchmark programs is also measured. Energy efficiency of a benchmark program is given by:

$$\text{Energy efficiency} = \frac{1/(\text{Execution time})}{\text{Power in watts}}$$

$$= \text{Program units/joule}$$

Energy Efficiency

- Efficiency averaged on *n* benchmark programs:

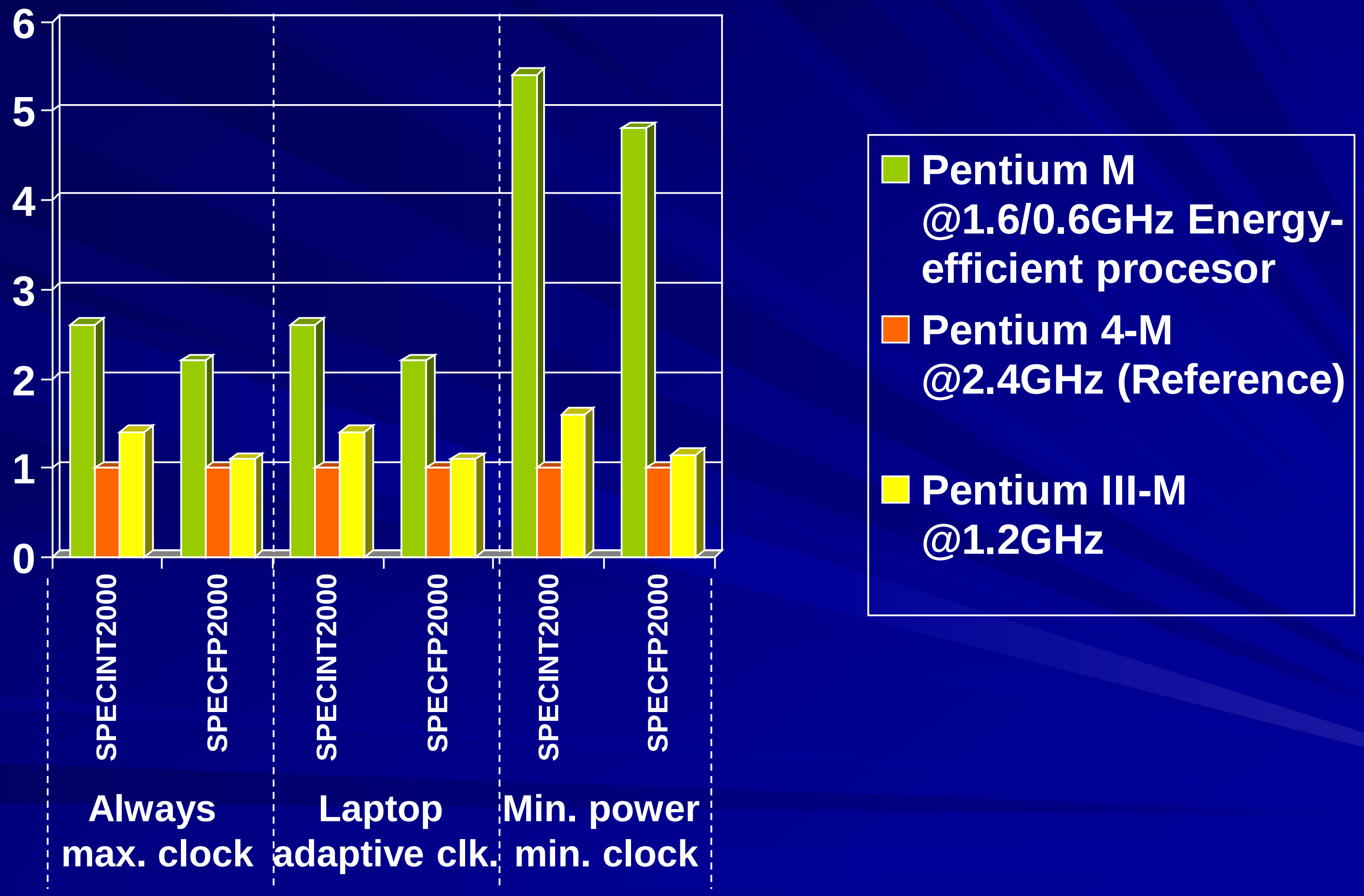
$$\text{Efficiency} = \left(\prod_{i=1}^{n} \text{Efficiency}_i\right)^{1/n}$$

where Efficiency_i is the efficiency for program *i*.

- Relative efficiency:

$$\text{Relative efficiency} = \frac{\text{Efficiency of a computer}}{\text{Eff. of reference computer}}$$

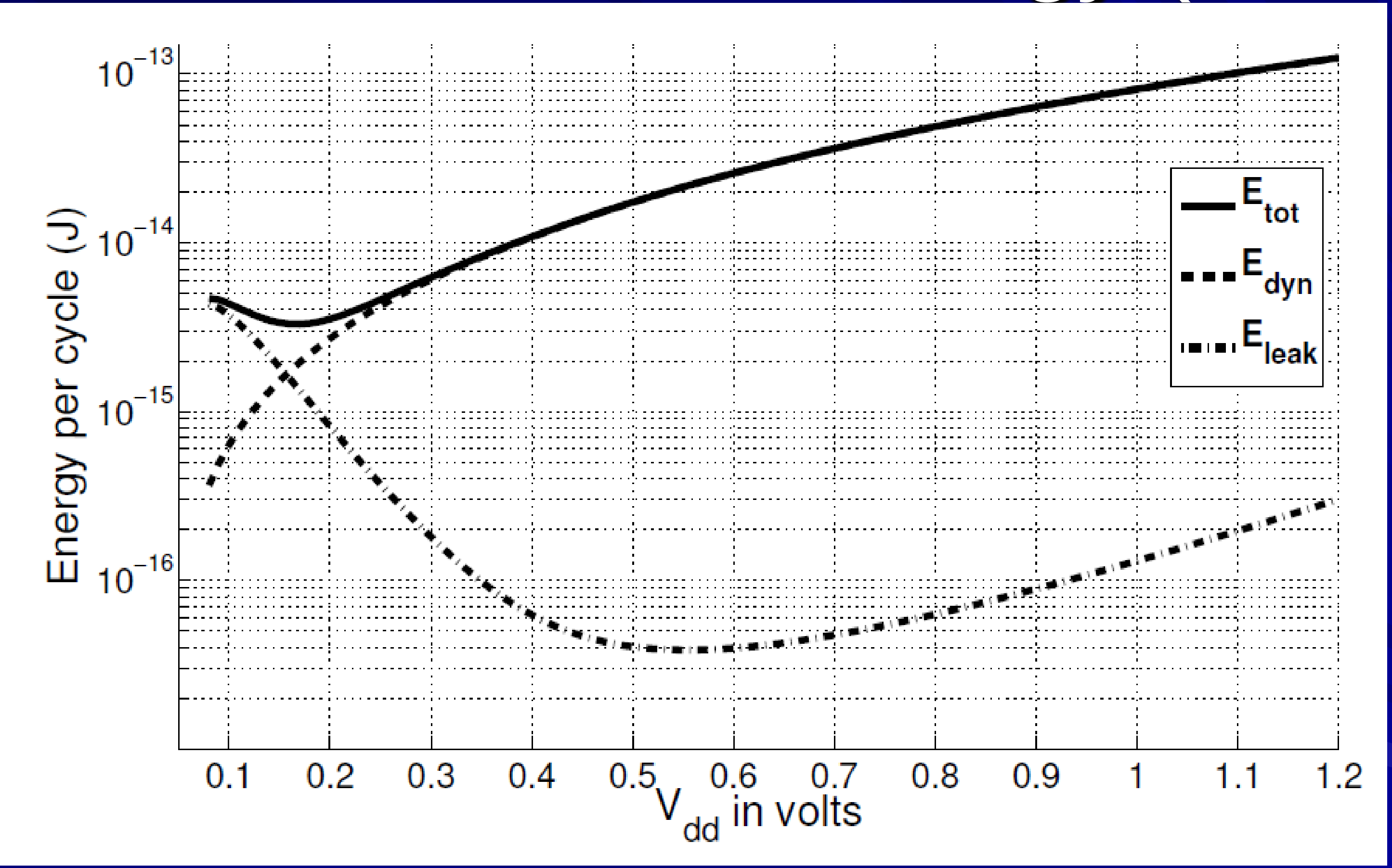
SPEC2000 Relative Energy Efficiency



Energy and Time Perspectives

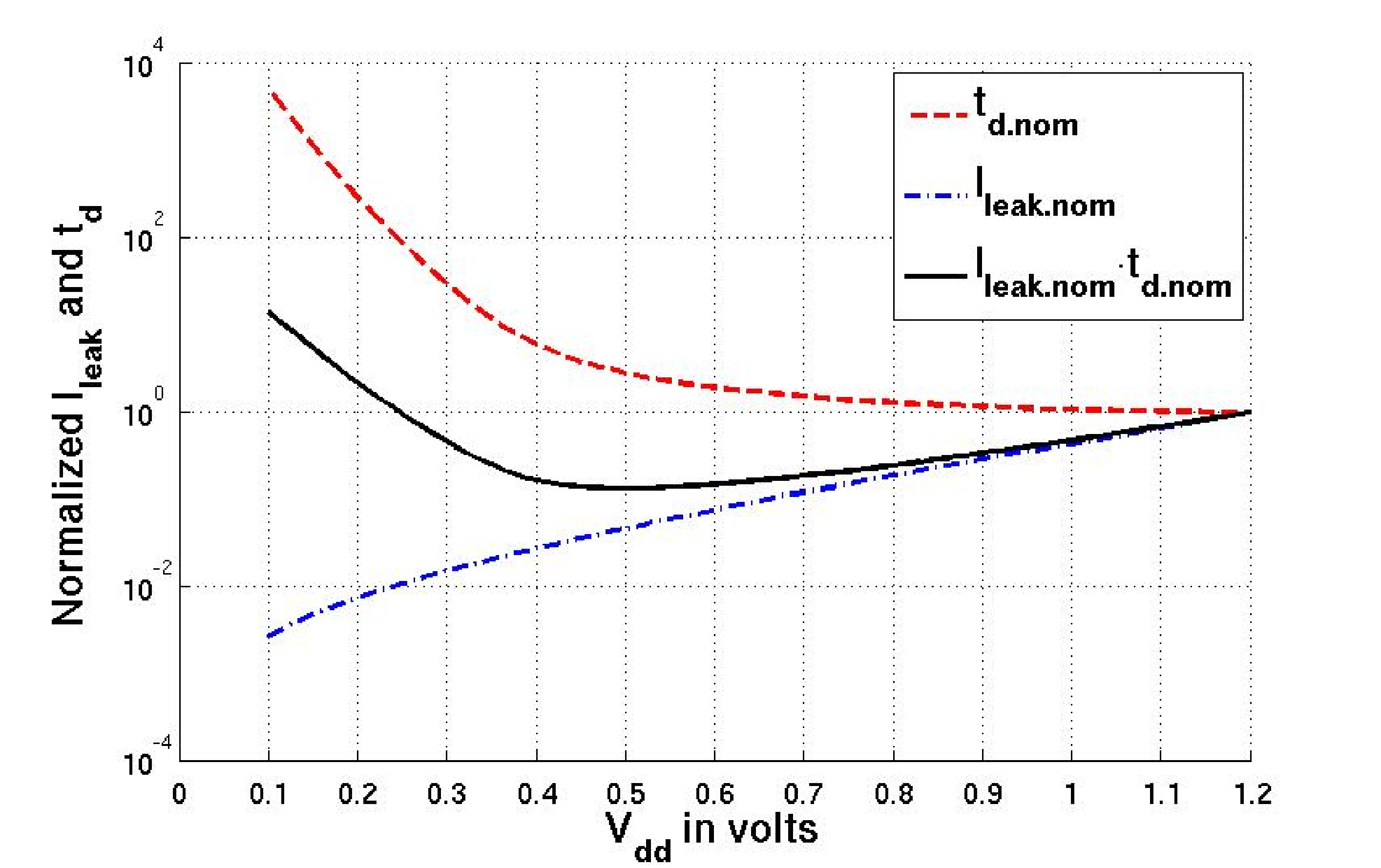
- Clock cycle is the unit of computing work.
- Cycle rate, f cycles per second
  - f is the rate of doing computing work
  - Hardware speed, similar to mph for a car
- Cycle efficiency, η cycles per joule
  - η is the computing work per energy unit
  - Hardware efficiency, similar to mpg for a car
- Results from recent work:
  - A. Shinde, "Managing Performance and Efficiency of a Processor," MEE Project Report, Auburn Univ., Dec. 2012.
  - A. Shinde and V. D. Agarwal, "Managing Performance and Efficiency of a Processor," *Proc. 45th IEEE Southeastern Symposium on System Theory*, Baylor Univ., TX, March 2013.

Energy/Cycle for an 8-bit Adder in 90nm CMOS Technology (PTM)



K. Kim, "Ultra Low Power CMOS Design" *PhD Dissertation*, Auburn University, Dept. of ECE, Auburn, Alabama, May 2011.

Delay of an 8-bit Adder in 90nm CMOS Technology (PTM)

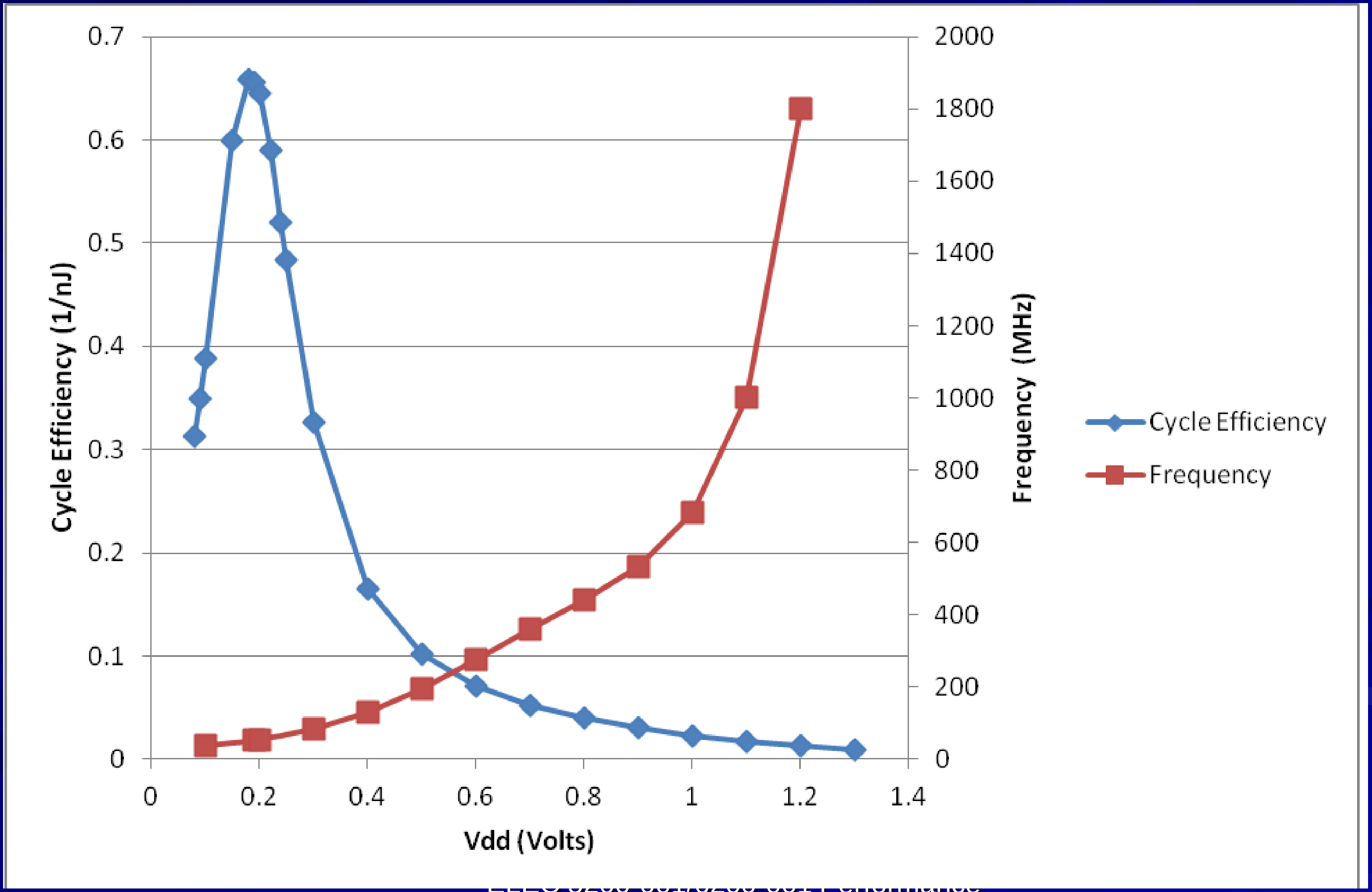


K. Kim, "Ultra Low Power CMOS Design" ***PhD Dissertation*****, Auburn University, Dept. of ECE, Auburn, Alabama, May 2011.**

Pentium M Processor

- Published data: H. Hanson, K. Rajamani, S. Keckler, F. Rawson, S. Ghiasi, J. Rubio, "Thermal Response to DVFS: Analysis with an Intel Pentium M," *Proc. International Symp. Low Power Electronics and Design*, 2007, pp. 219-224.
- VDD = 1.2V
- Maximum clock rate = 1.8GHz
- Critical path delay, td = 1/1.8GHz = 555.56ps
- Power consumption = 120W
- Energy per cycle, EPC = 120/(1.8GHz) = 66.67nJ

Cycle Efficiency and Frequency for Pentium M



Example of Power Management

- For a program that executes in 1.8 billion clock cycles.

Voltage VDD	Frequency f MHz	Cycle Efficiency, η	Execution Time second	Total Energy Consumed	Power f/η
1.2 V	1800 megacycles/s	15 megacycles/joule	1.0	120 Joules	120W
0.6 V	277 megacycles/s	70 megacycles/joule	6.5	25 Joules	39.6W
200 mV	54.5 megacycles/s	660 megacycles/joule	33	2.72 Joules	0.083W

Ways of Improving Performance

- Increase clock rate.
- Improve processor organization for lower CPI
  - Pipelining
  - Instruction-level parallelism (ILP): MIMD (Scalar)
  - Data-parallelism: SIMD (Vector)
  - multiprocessing
- Compiler enhancements that lower the instruction count or generate instructions with lower average CPI (e.g., by using simpler instructions).

Limits of Performance

- Execution time of a program on a computer is 100 s:
  - 80 s for multiply operations
  - 20 s for other operations
- Improve multiply *n* times:

Execution time = $\left(\frac{80}{n} + 20\right)$ seconds

- Limit: Even if $n = \infty$, execution time cannot be reduced below 20 s.

Amdahl's Law

- The execution time of a system, in general, has two fractions – a fraction f_{enh} that can be speeded up by factor n, and the remaining fraction $1 - f_{enh}$ that cannot be improved. Thus, the possible speedup is:

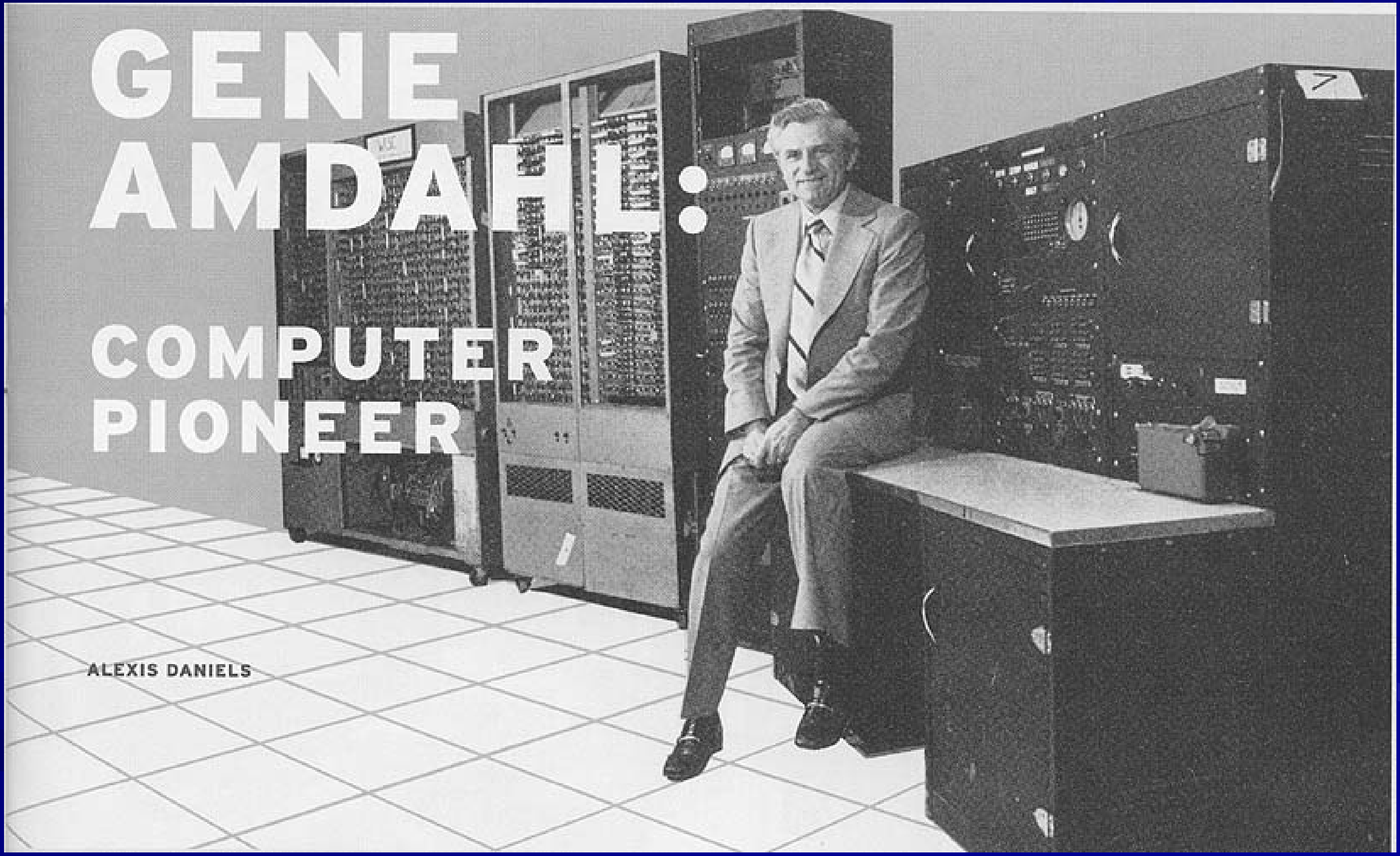
$$\textit{Speedup} = \frac{\textit{Old time}}{\textit{New time}} = \frac{1}{1 - f_{enh} + f_{enh}/n}$$

- G. M. Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," *Proc. AFIPS Spring Joint Computer Conf.*, Atlantic City, NJ, April 1967, pp. 483-485.

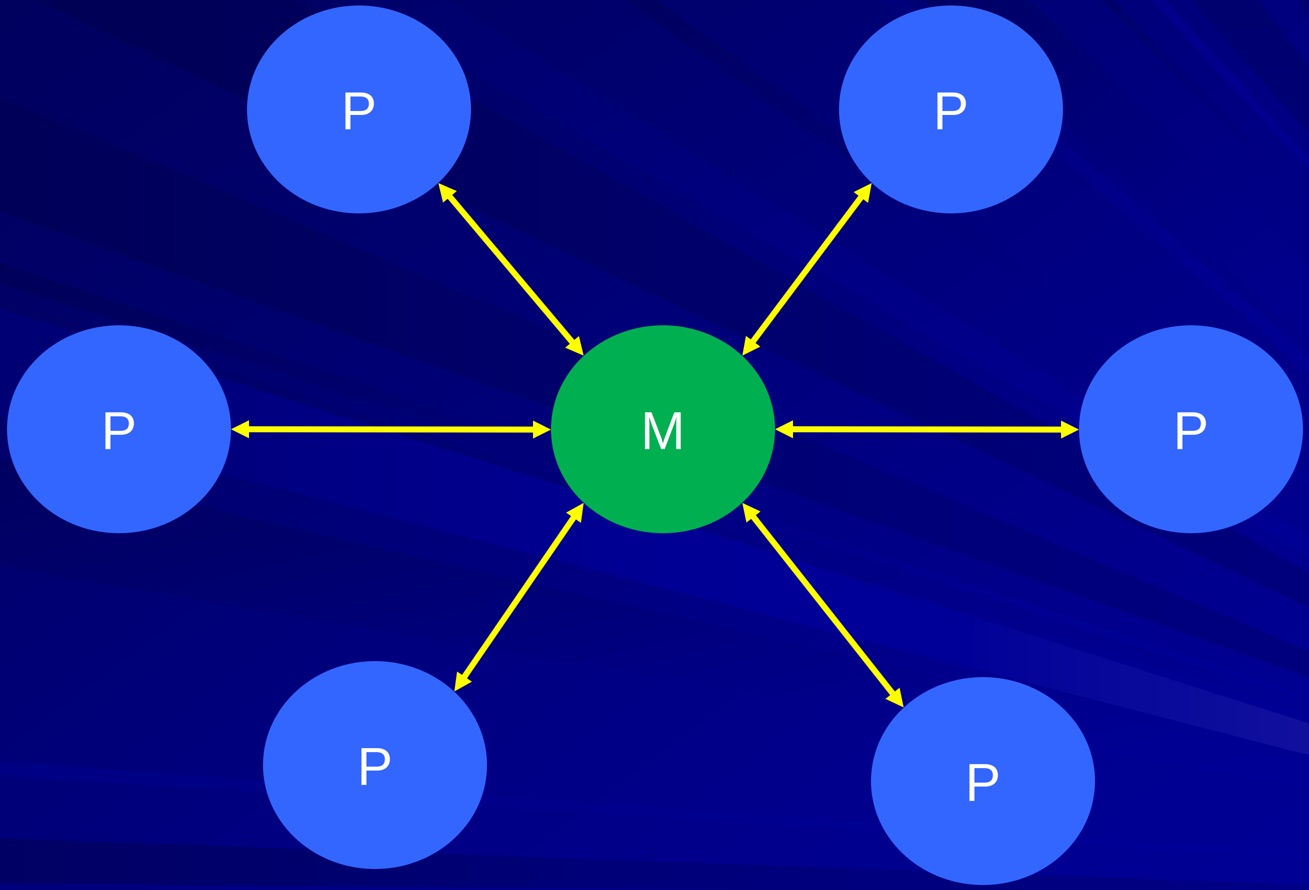
Gene Myron Amdahl born 1922

http://en.wikipedia.org/wiki/Gene_Amdahl

Wisconsin Integrally Synchronized Computer (WISC), 1950-51



Parallel Processors: Shared Memory

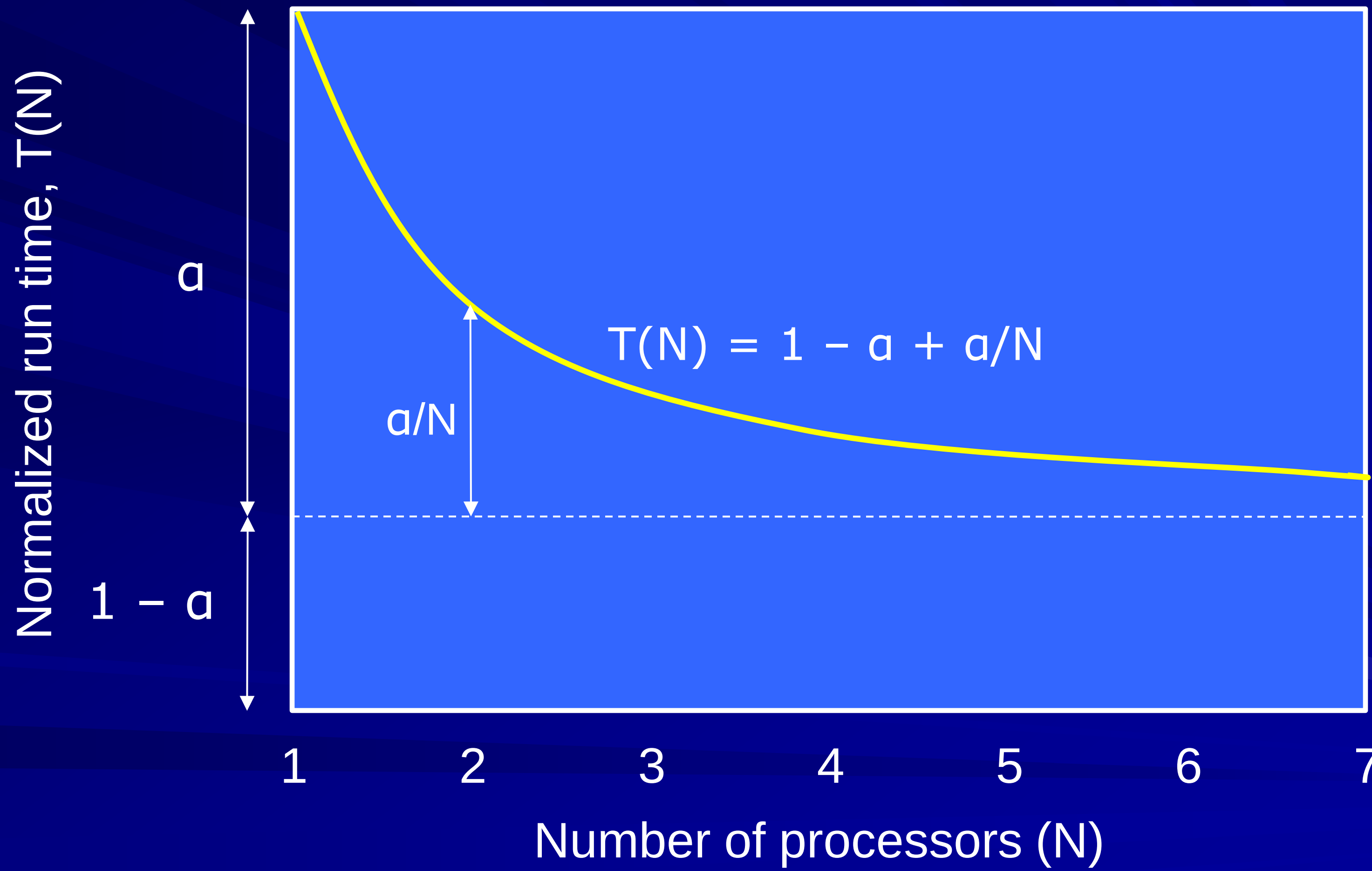


Parallel Processors
Shared Memory, Infinite Bandwidth

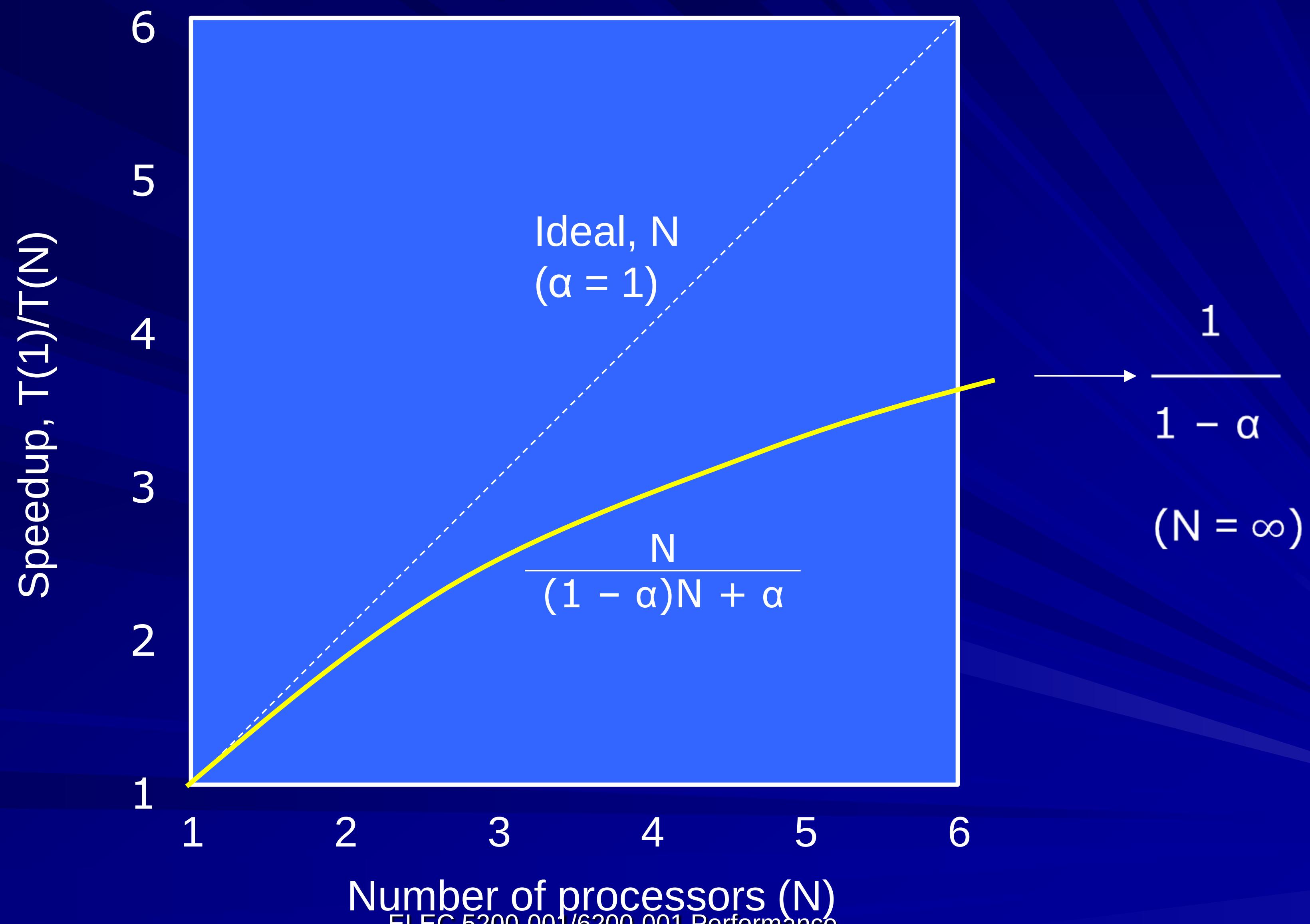
- N processors
- Single processor: non-memory execution time = α
- Memory access time = $1 - \alpha$
- N processor run time, $T(N) = 1 - \alpha + \alpha/N$
- Speedup $= \dfrac{T(1)}{T(N)} = \dfrac{1}{1 - \alpha + \alpha/N} = \dfrac{N}{(1 - \alpha)N + \alpha}$

Maximum speedup $= 1/(1 - \alpha)$, when $N = \infty$

Run Time


Normalized run time, T(N)
α
1 − α
α/N
T(N) = 1 − α + α/N
1
2
3
4
5
6
7
Number of processors (N)

Speedup


6
5
4
3
2
1
1
2
3
4
5
6
Speedup, T(1)/T(N)
Number of processors (N)
Ideal, N
(α = 1)
N
(1 − α)N + α
1
1 − α
(N = ∞)

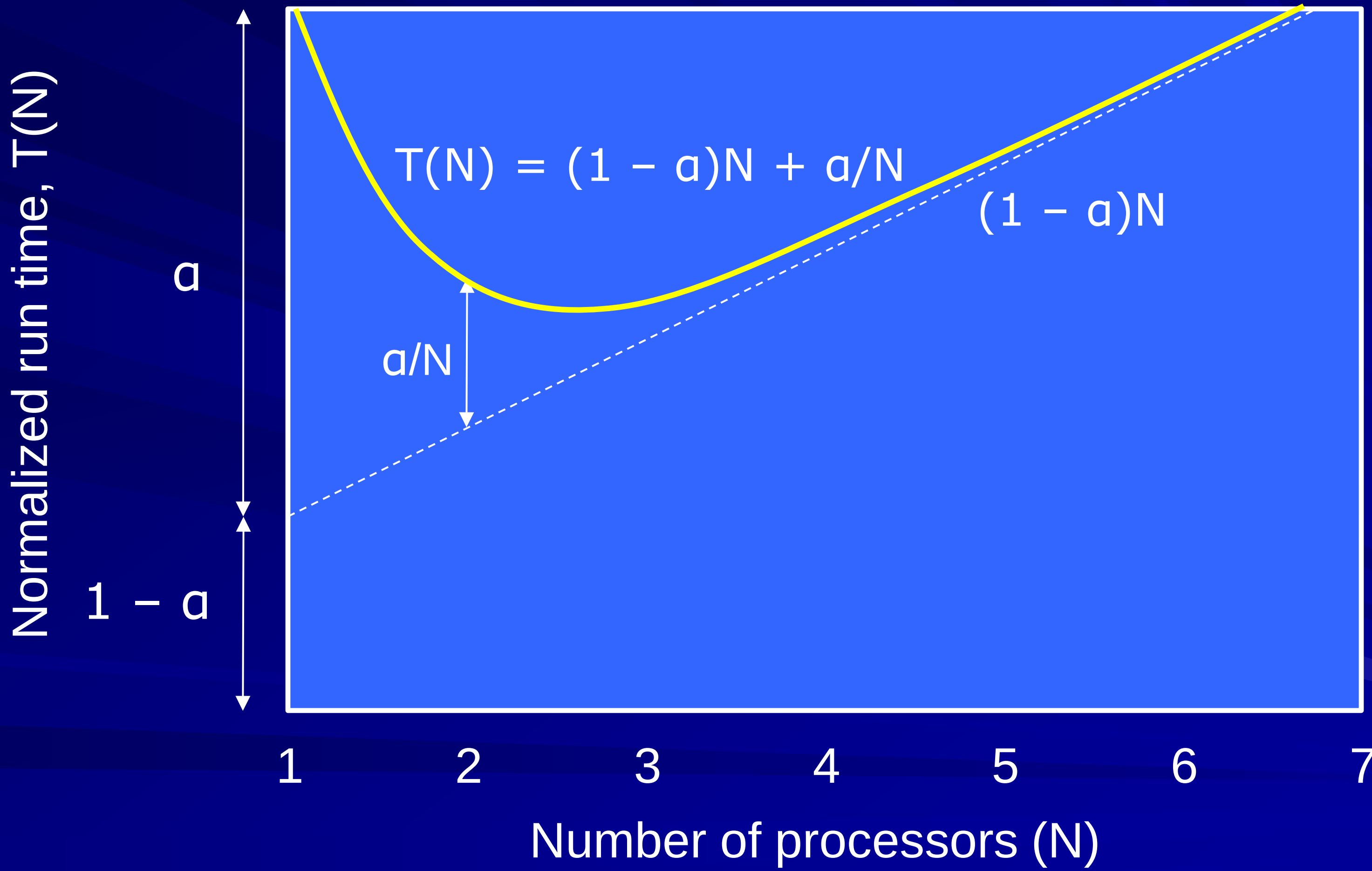
Example

- 10% memory accesses, i.e., $\alpha = 0.9$
- Maximum speedup $= 1/(1 - a)$
  $= 1.0/0.1 = 10,$
  when $N = \infty$
- What is the speedup with 10 processors?

Parallel Processors
Shared Memory, Finite Bandwidth

- N processors
- Single processor: non-memory execution time = α
- Memory access time = $(1 - \alpha)N$
- N processor run time, $T(N) = (1 - \alpha)N + \alpha/N$
- Speedup $= \dfrac{1}{(1 - \alpha)N + \alpha/N} = \dfrac{N}{(1 - \alpha)N^2 + \alpha}$

Run Time



T(N) = (1 – α)N + α/N
(1 – α)N
α/N
α
1 – α
Normalized run time, T(N)
1
2
3
4
5
6
7
Number of processors (N)

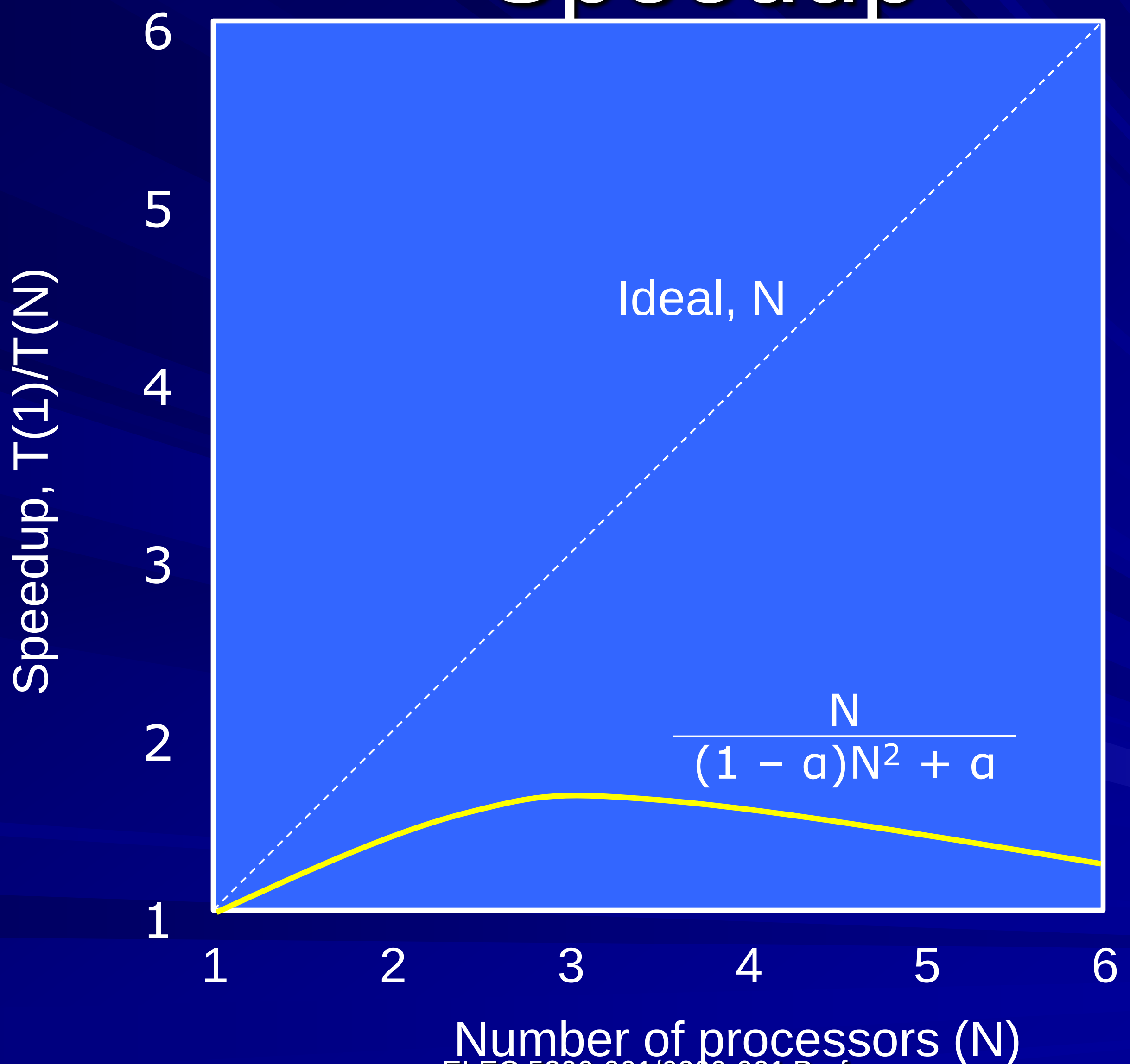
Minimum Run Time

- Minimize N processor run time,

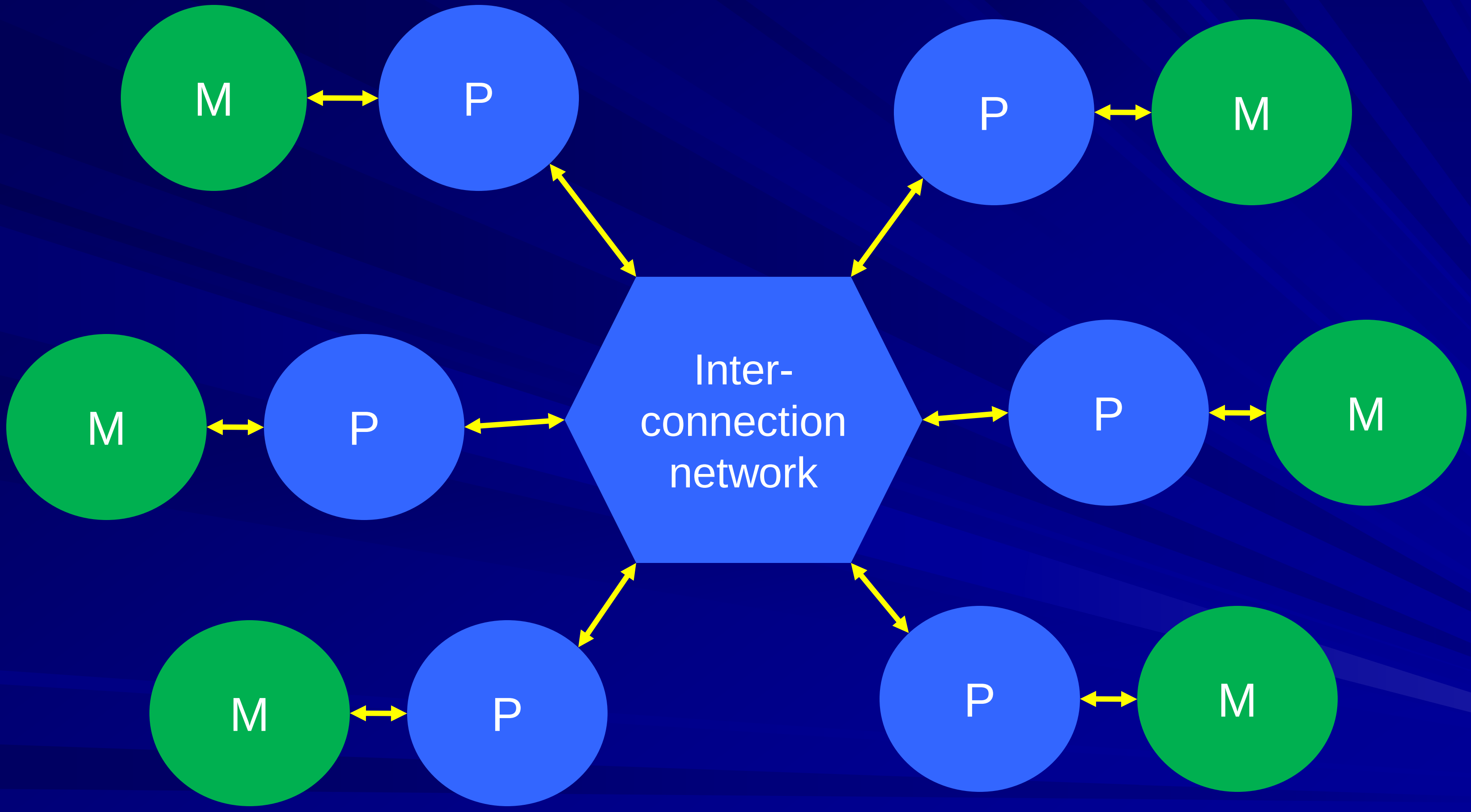
$$T(N) = (1 - \alpha)N + \alpha/N$$

- $\partial T(N)/\partial N = 0$
- $1 - \alpha - \alpha/N^2 = 0$, $N = [\alpha/(1 - \alpha)]^{½}$
- Min. $T(N) = 2[\alpha(1 - \alpha)]^{½}$, because $\partial^2 T(N)/\partial N^2 > 0$.
- Maximum speedup = $1/T(N) = 0.5[\alpha(1 - \alpha)]^{-½}$
- Example: $\alpha = 0.9$
  - Maximum speedup = 1.67, when N = 3

Speedup


Speedup, T(1)/T(N)
6
5
4
3
2
1
Ideal, N
$\frac{N}{(1-\alpha)N^2+\alpha}$
1
2
3
4
5
6
Number of processors (N)

Parallel Processors: Distributed Memory



Parallel Processors Distributed Memory

- N processors
- Single processor: non-memory execution time = α
- Memory access time = 1 – α, same as single processor
- Communication overhead = β(N – 1)
- N processor run time, T(N) = β(N – 1) + 1/N
- Speedup $= \dfrac{1}{\beta(N-1) + 1/N} = \dfrac{N}{\beta N(N-1) + 1}$

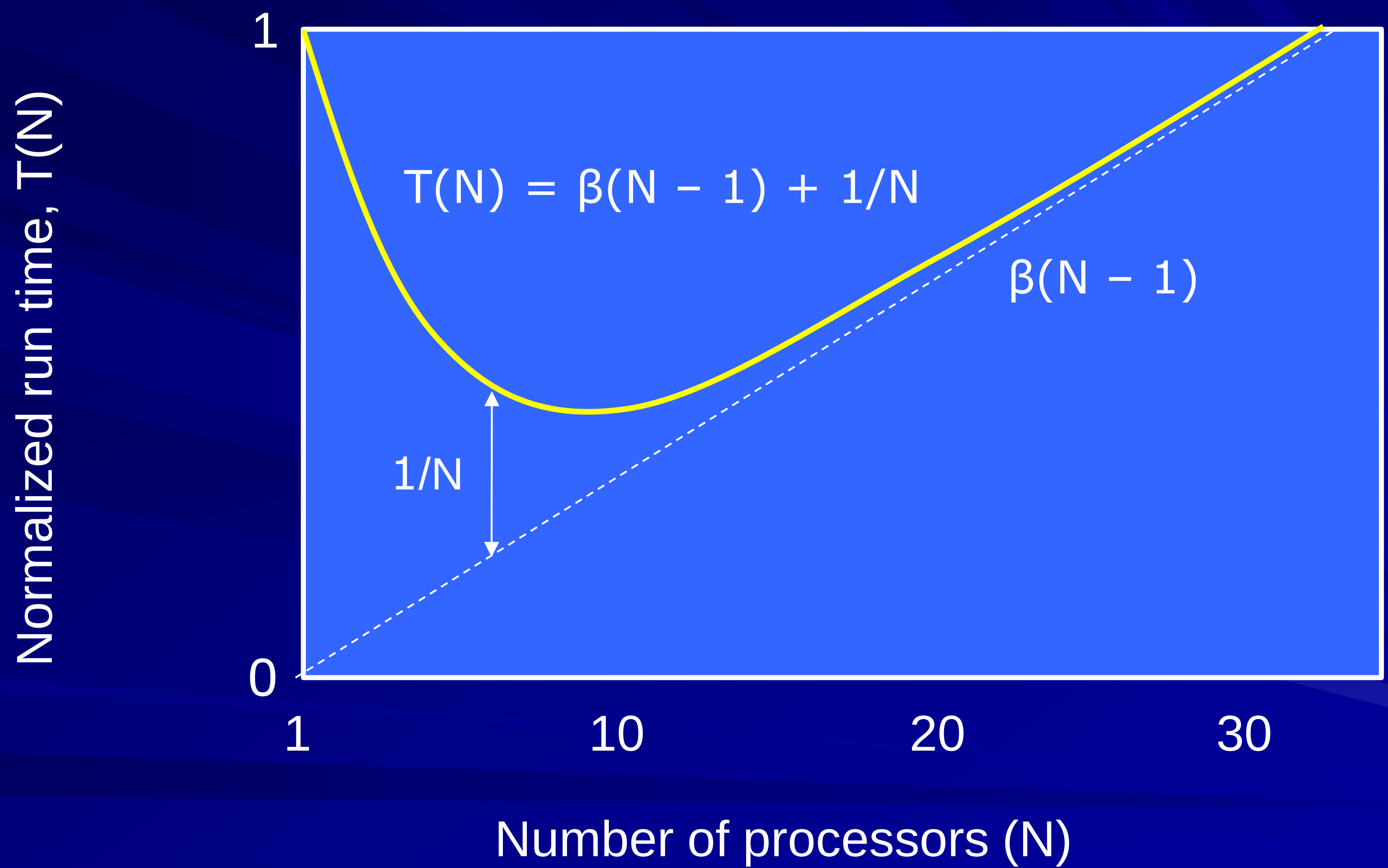
Minimum Run Time

- Minimize N processor run time,

$$T(N) = \beta(N - 1) + 1/N$$

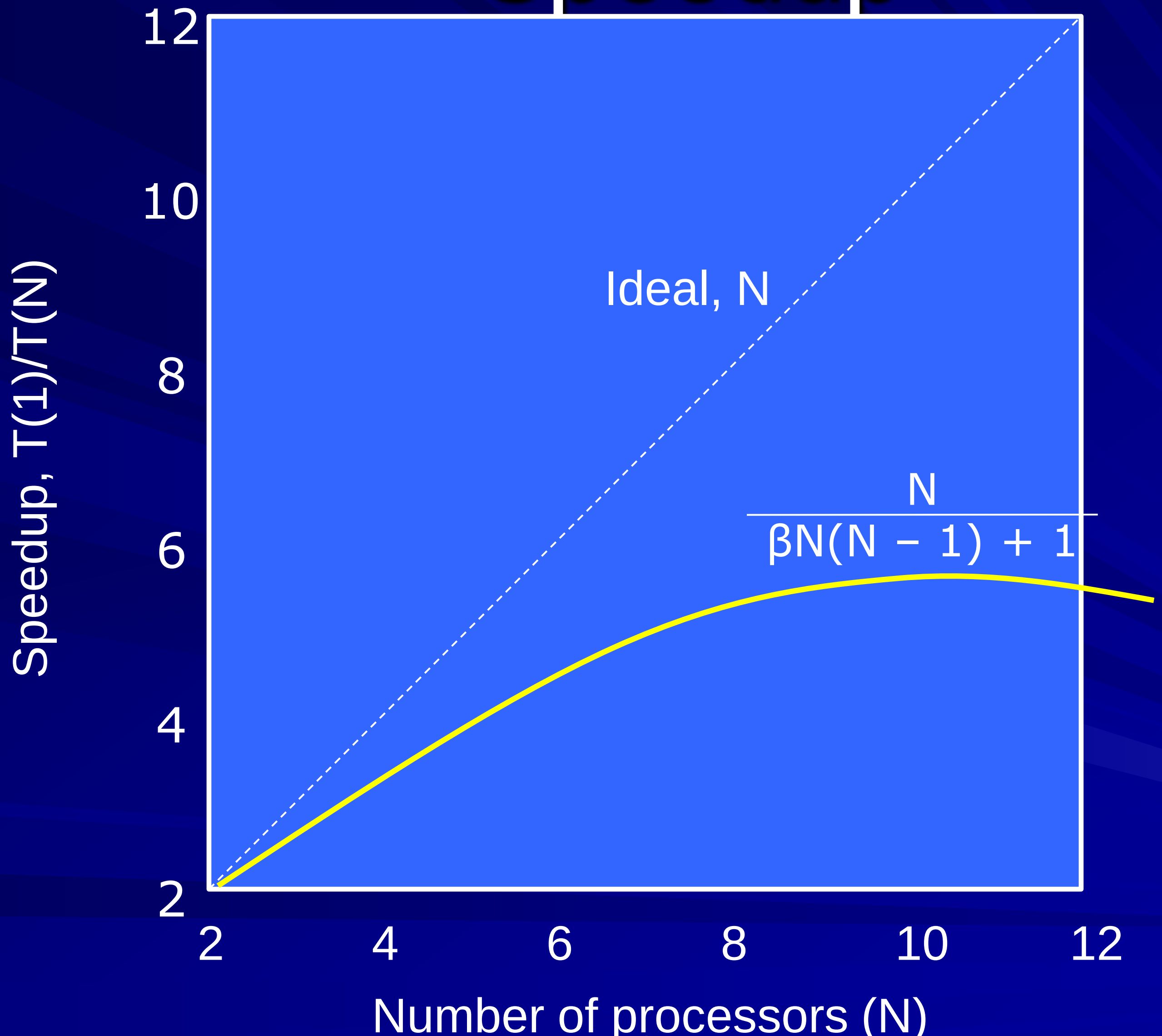
- $\partial T(N)/\partial N = 0$
- $\beta - 1/N^2 = 0$, $N = \beta^{-1/2}$
- Min. $T(N) = 2\beta^{1/2} - \beta$, because $\partial^2 T(N)/\partial N^2 > 0$.
- Maximum speedup = $1/T(N) = 1/(2\beta^{1/2} - \beta)$
- Example: $\beta = 0.01$, Maximum speedup:
  - $N = 10$
  - $T(N) = 0.19$
  - Speedup = 5.26

Run Time



1
0
T(N) = β(N – 1) + 1/N
β(N – 1)
1/N
1
10
20
30
Normalized run time, T(N)
Number of processors (N)

Speedup


12
10
8
6
4
2
Ideal, N
N
βN(N – 1) + 1
2
4
6
8
10
12
Speedup, T(1)/T(N)
Number of processors (N)

Further Reading

- G. M. Amdahl, “Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities,” *Proc. AFIPS Spring Joint Computer Conf*., Atlantic City, NJ, Apr. 1967, pp. 483-485.
- J. L. Gustafson, “Reevaluating Amdahl’s Law,” *Comm. ACM*, vol. 31, no. 5, pp. 532-533, May 1988.
- M. D. Hill and M. R. Marty, “Amdahl’s Law in the Multicore Era,” *Computer*, vol. 41, no. 7, pp. 33-38, July 2008.
- D. H. Woo and H.-H. S. Lee, “Extending Amdahl’s Law for Energy-Efficient Computing in the Many-Core Era,” *Computer*, vol. 41, no. 12, pp. 24-31, Dec. 2008.
- S. M. Pieper, J. M. Paul and M. J. Schulte, “A New Era of Performance Evaluation,” *Computer*, vol. 40, no. 9, pp. 23-30, Sep. 2007.
- S. Gal-On and M. Levy, “Measuring Multicore Performance,” *Computer*, vol. 41, no. 11, pp. 99-102, November 2008.
- S. Williams, A. Waterman and D. Patterson, “Roofline: An Insightful Visual Performance Model for Multicore Architectures,” *Comm. ACM*, vol. 52, no. 4, pp. 65-76, Apr. 2009.