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

Spring 2019

Performance of a Computer

Christopher B. Harris
Assistant Professor
Department of Electrical and Computer Engineering
Auburn University, Auburn, AL 36849

(Adapted from slides by Vishwani D. Agrawal)

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.

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: *nix “time” Command

90.7s	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

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

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

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

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.
- Coremark benchmark
- 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.

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.

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

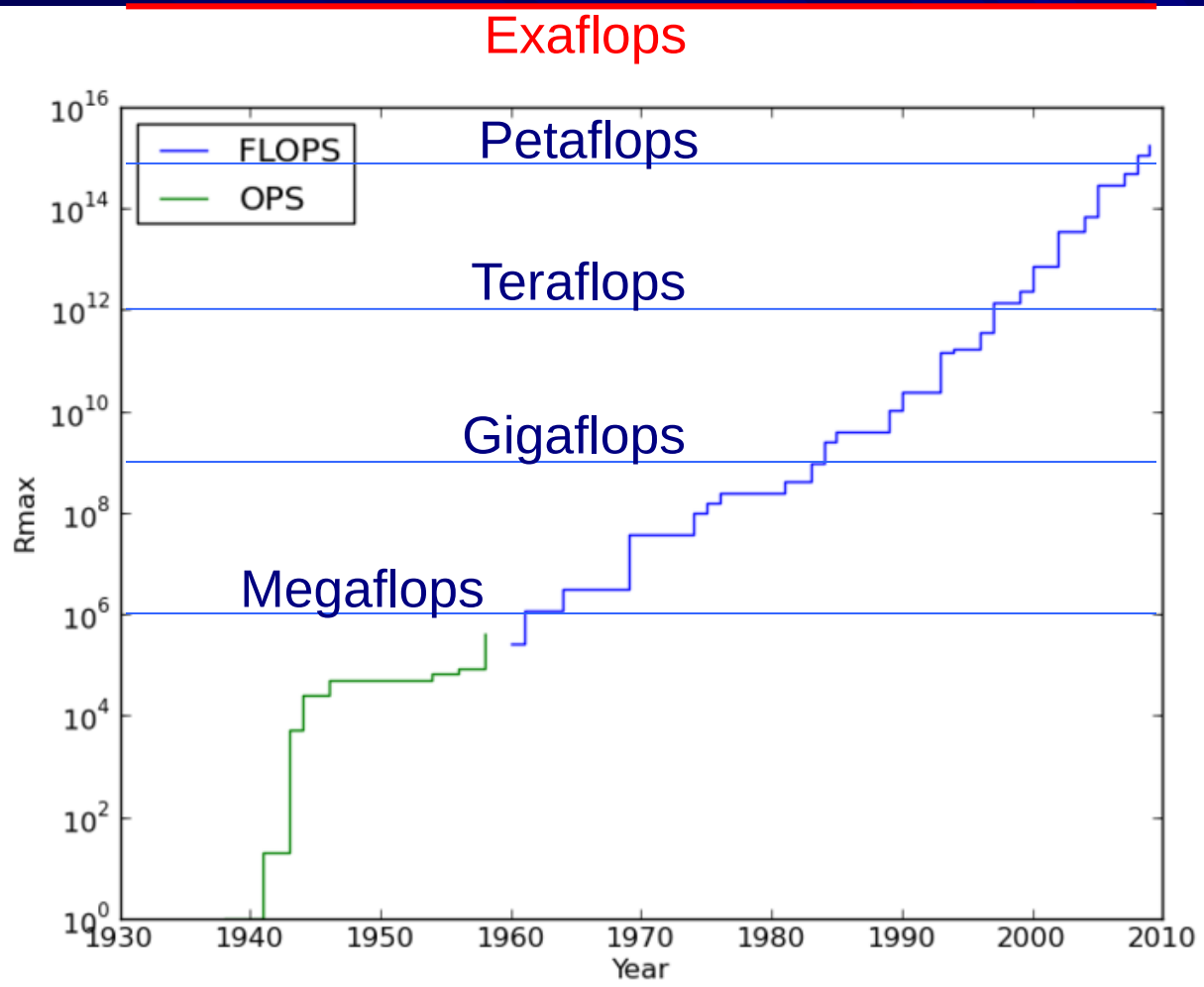
New York Times, April 20, 1994

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, Nov 2017

www.top500.org

Rank	Name	Location	Cores	Clock GHz	Max. Pflops	Power MW	η Pflops /MJoule
1	Sunway TaihuLight	Wuxi China	10,649,600	1.45	125.43	15.37	8.16
2	Tianhe-2	Guangzhou China	3,120,000	2.2	33.86	17.80	1.90
3	Piz Daint	Lugano Switzerland	361,760	2.6	25.32	2.27	11.15
4	Gyoukou	Yokusuka Japan	19,860,000	1.3	19.14	1.35	14.18
5	Titan	Tenn. USA	560,460	2.2	17.59	8.21	2.14

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

Performance

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

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

or

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

- Performance is inverse of execution time:

$$Performance = 1/(Execution\ time)$$

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

$$\neq \frac{\{T1 + \ldots + Tn\}/n}{\{r1 + \ldots + rn\}/n}$$

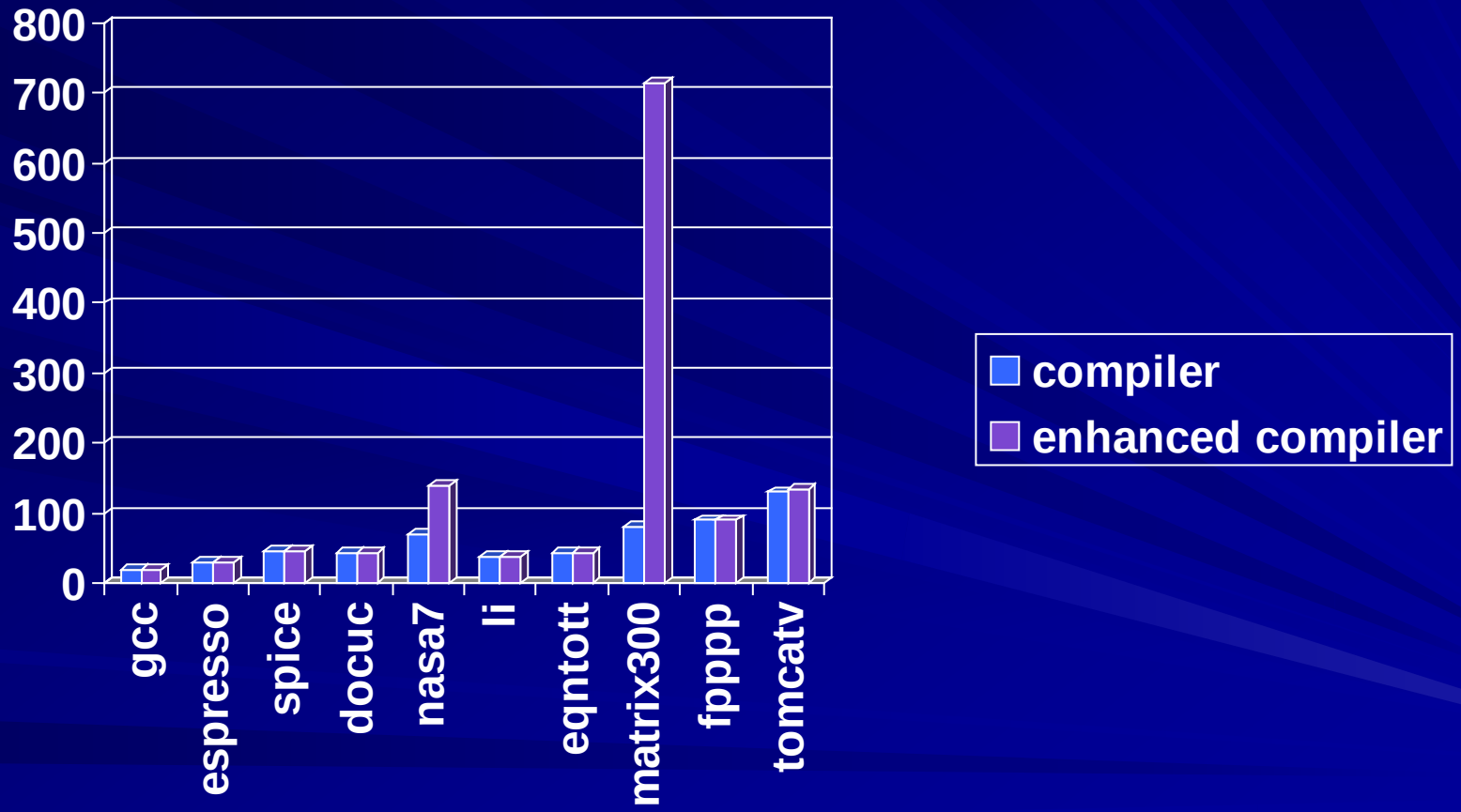
Not used

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.
- Retired in 2007, replaced with SPEC CPU™ 2006 https://www.spec.org/cpu2006/

CINT2000 : Eleven Programs

Name	Ref Time	Remarks
164.gzip	1400	Data compression utility (C)
175.vpr	1400	FPGA circuit placement and routing (C)
176.gcc	1100	C compiler (C)
181.mcf	1800	Minimum cost network flow solver (C)
186.crafty	1000	Chess program (C)
197.parser	1800	Natural language processing (C)
252.eon	1300	Ray tracing (C++)
253.perlbmk	1800	Perl (C)
254.gap	1100	Computational group theory (C)
255.vortex	1900	Object Oriented Database (C)
256.bzip2	1500	Data compression utility (C)
300.twolf	3000	Place and route simulator (C)

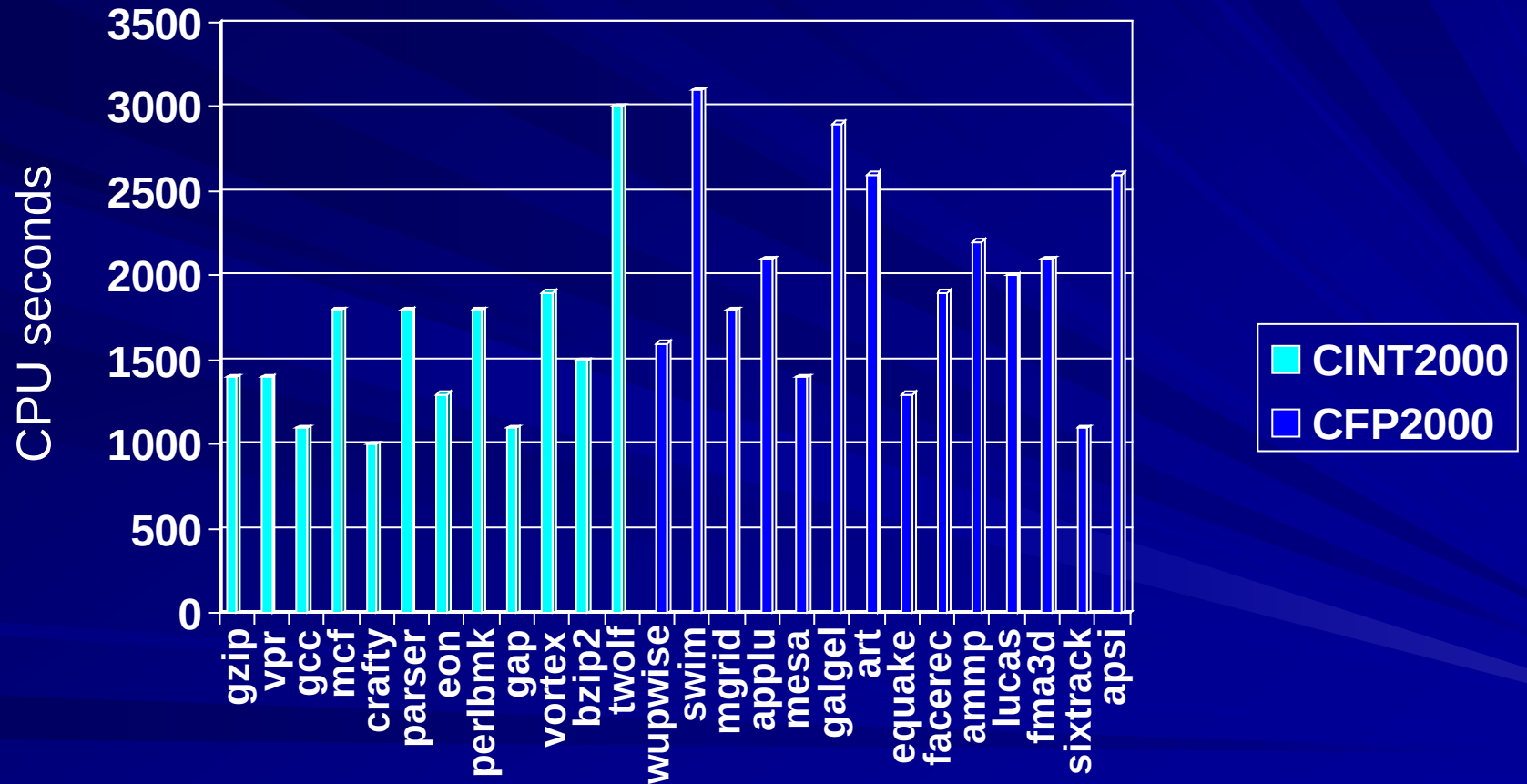
https://www.spec.org/cpu2000/docs/readme1st.html#Q8

CFP2000: Fourteen Programs
(6 Fortran 77, 4 Fortran 90, 4 C)

Name	Ref Time	Remarks
168.wupwise	1600	Quantum chromodynamics
171.swim	3100	Shallow water modeling
172.mgrid	1800	Multi-grid solver in 3D potential field
173.applu	2100	Parabolic/elliptic partial differential equations
177.mesa	1400	3D Graphics library
178.galgel	2900	Fluid dynamics: analysis of oscillatory instability
179.art	2600	Neural network simulation; adaptive resonance theory
183.equake	1300	Finite element simulation; earthquake modeling
187.facerec	1900	Computer vision: recognizes faces
188.ammp	2200	Computational chemistry
189.lucas	2000	Number theory: primality testing
191.fma3d	2100	Finite element crash simulation
200.sixtrack	1100	Particle accelerator model
301.apsi	2600	Solves problems regarding temperature, wind, velocity and distribution of pollutants

https://www.spec.org/cpu2000/docs/readme1st.html#Q8

Reference CPU: Sun Ultra 5_10 300MHz Processor

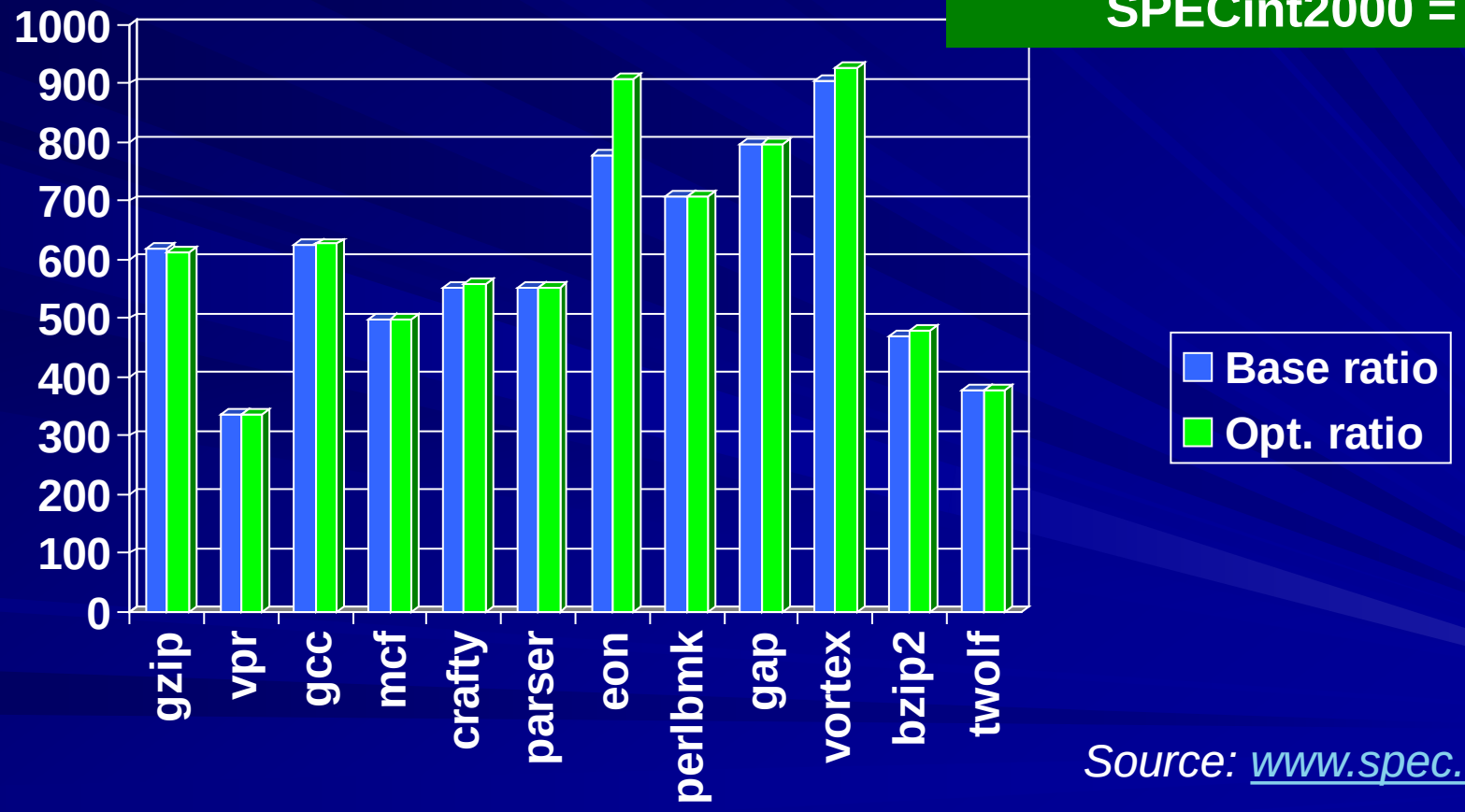


Two Benchmark Results

- 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.

CINT2000: 1.7GHz Pentium 4 (D850MD Motherboard)

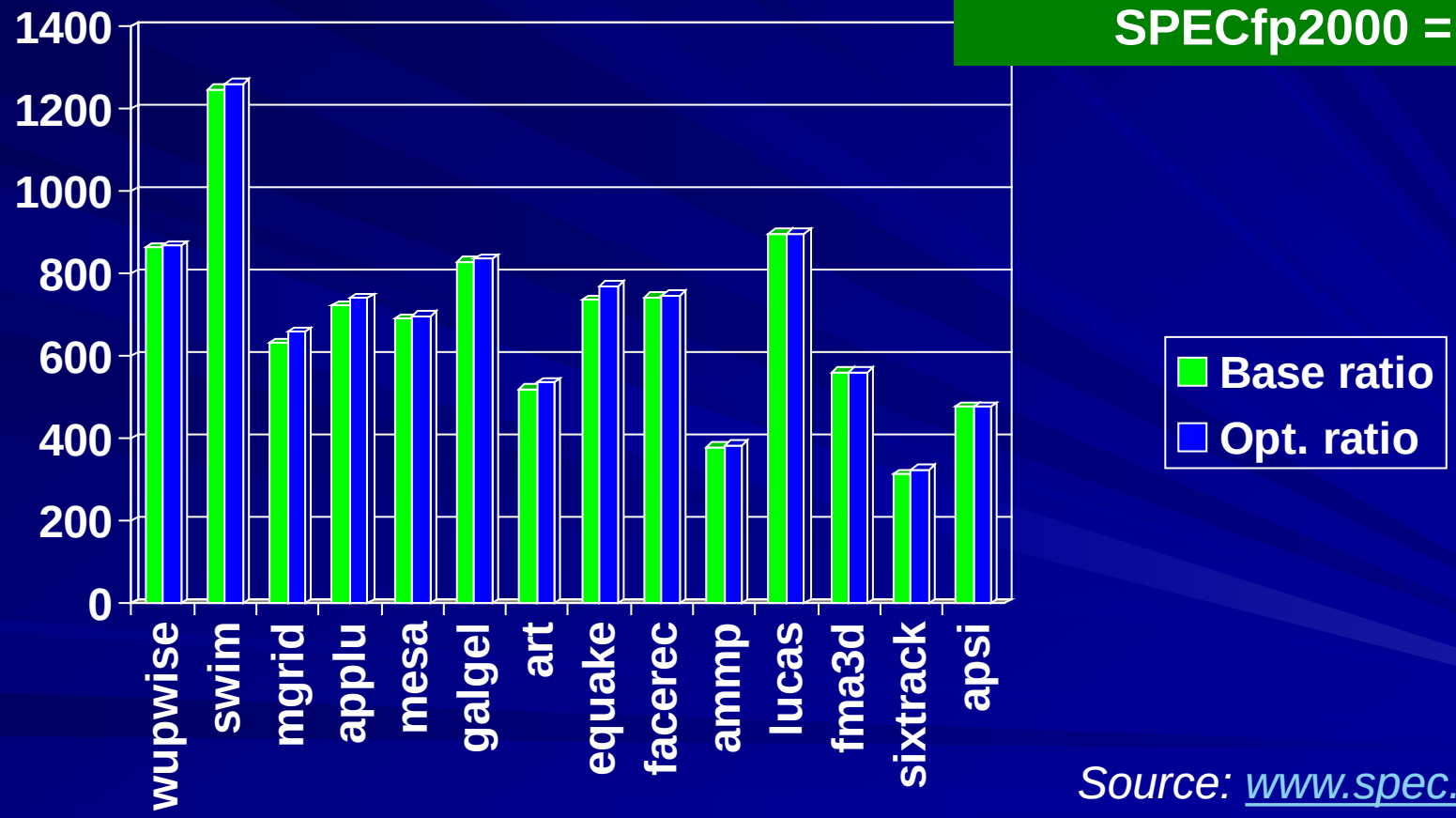
SPECint2000_base = 579
SPECint2000 = 588



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:

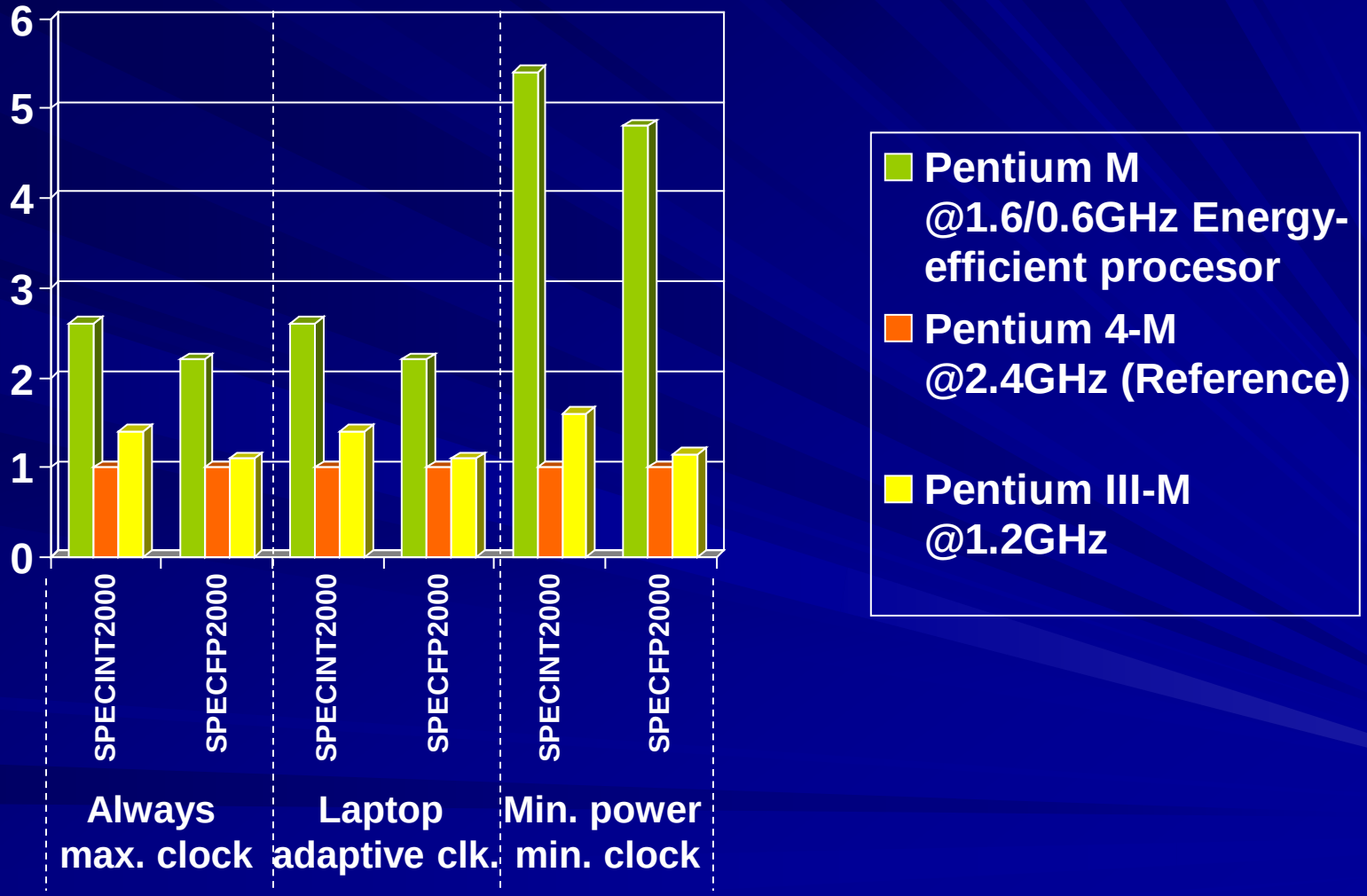
$$\text{Efficiency} = \left(\prod_{i=1}^{n} \text{Efficiency}_i \right)^{1/n}$$

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



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).

Floating-point performance

- Peak floating point performance
  - Sum of floating point performance for all cores on a chip.

Floating Point Performance Impact on Memory Subsystem

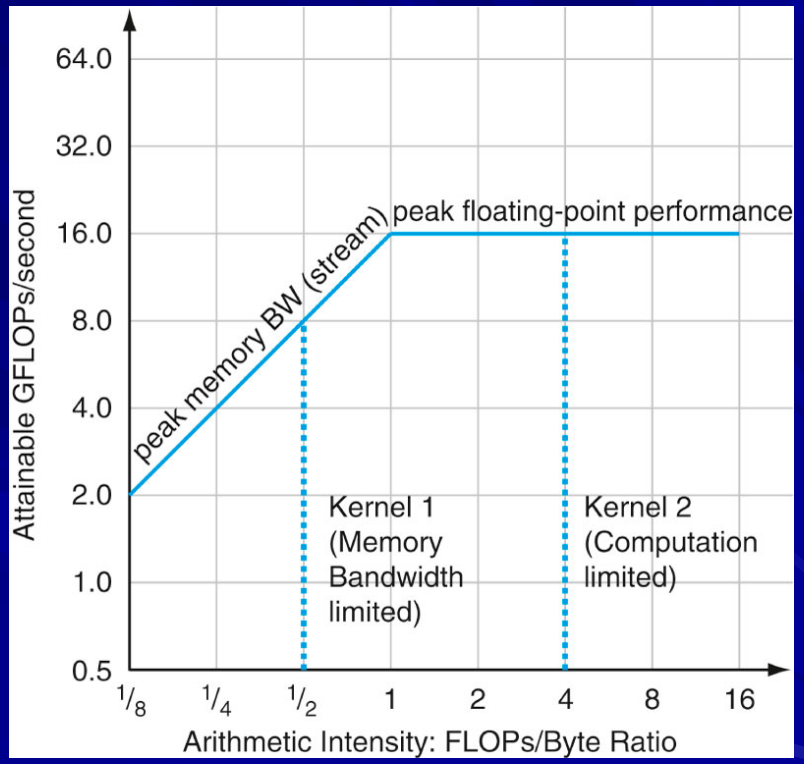
- Algorithmic intensity

$$\frac{\text{Floating-point ops / sec}}{\text{Floating-point ops / byte}} = \text{bytes / sec}$$

Peak floating point perf divided by average floating point operations per byte of memory accessed.

A New Performance Model

- Roofline Model
- Diagram is calculated once per machine.
- Can determine if a given program (kernel) is CPU or I/O limited.



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:

$$Speedup = \frac{Old\ time}{New\ time}$$

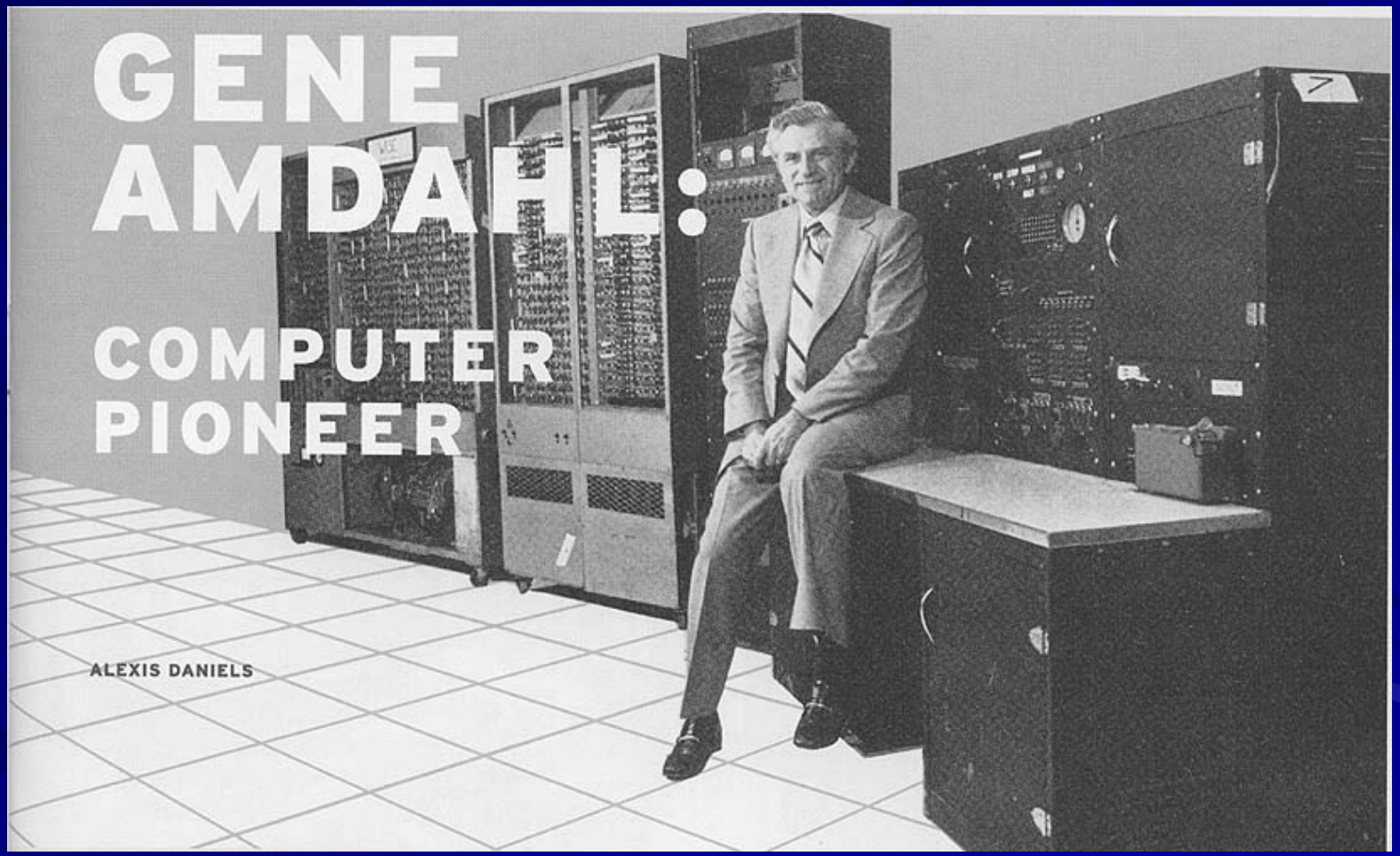
$$= \frac{1}{1 - f_{enh} + f_{enh}/n}$$

- 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.

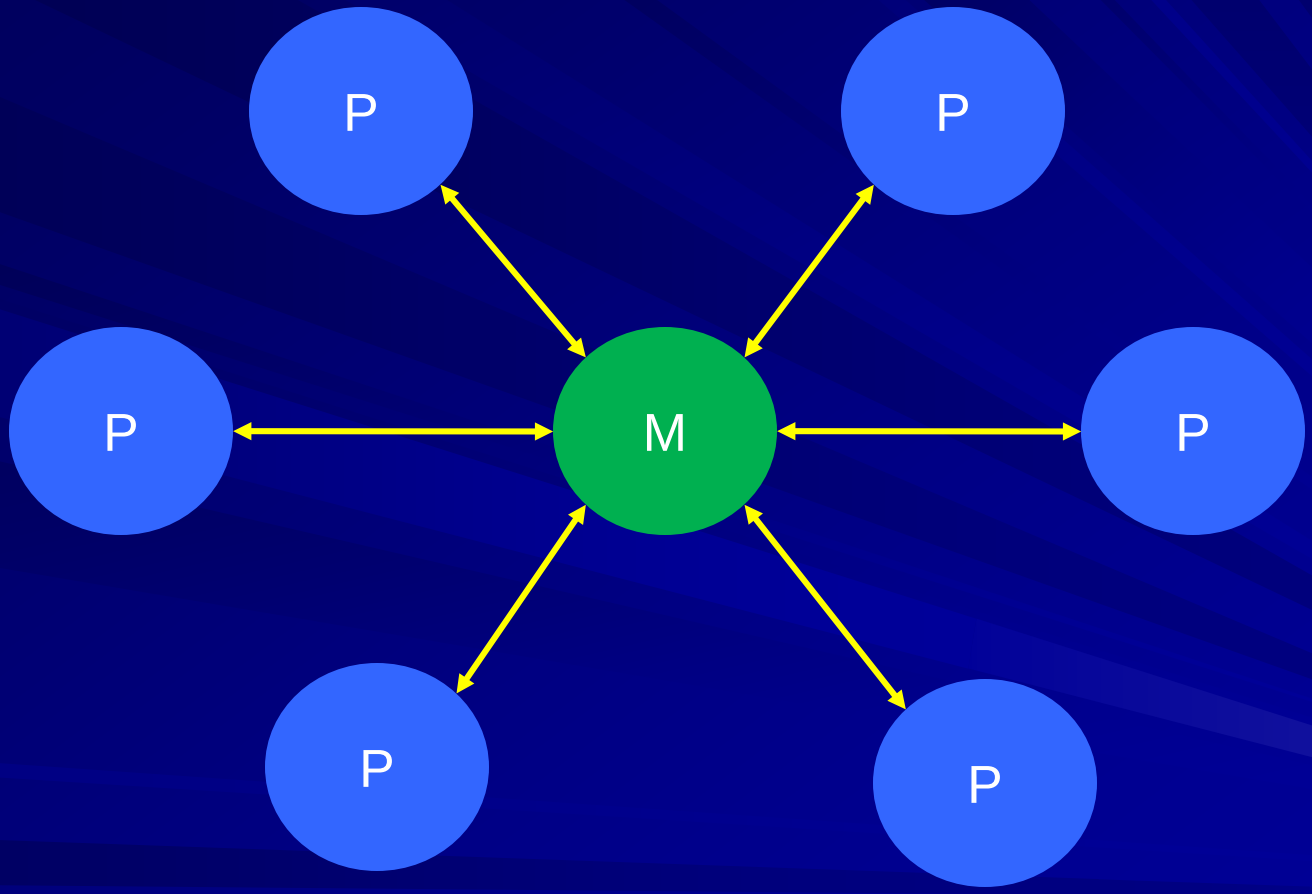
Gene Myron Amdahl
born 1922

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

Wisconsin Integrally Synchronized Computer (WISC), 1950-51



Parallel Processors: Shared Memory



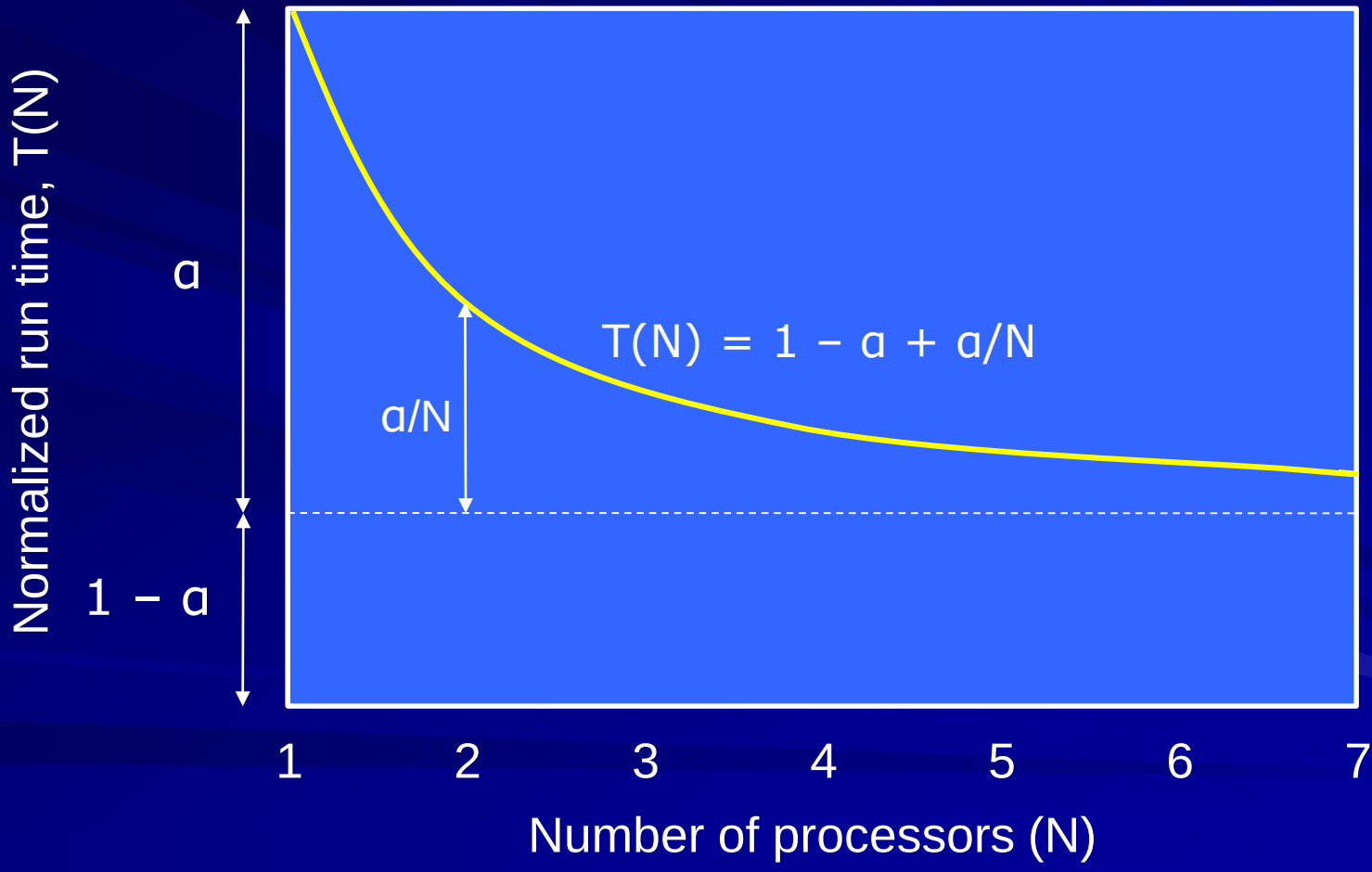
Parallel Processors
Shared Memory, Infinite Bandwidth

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

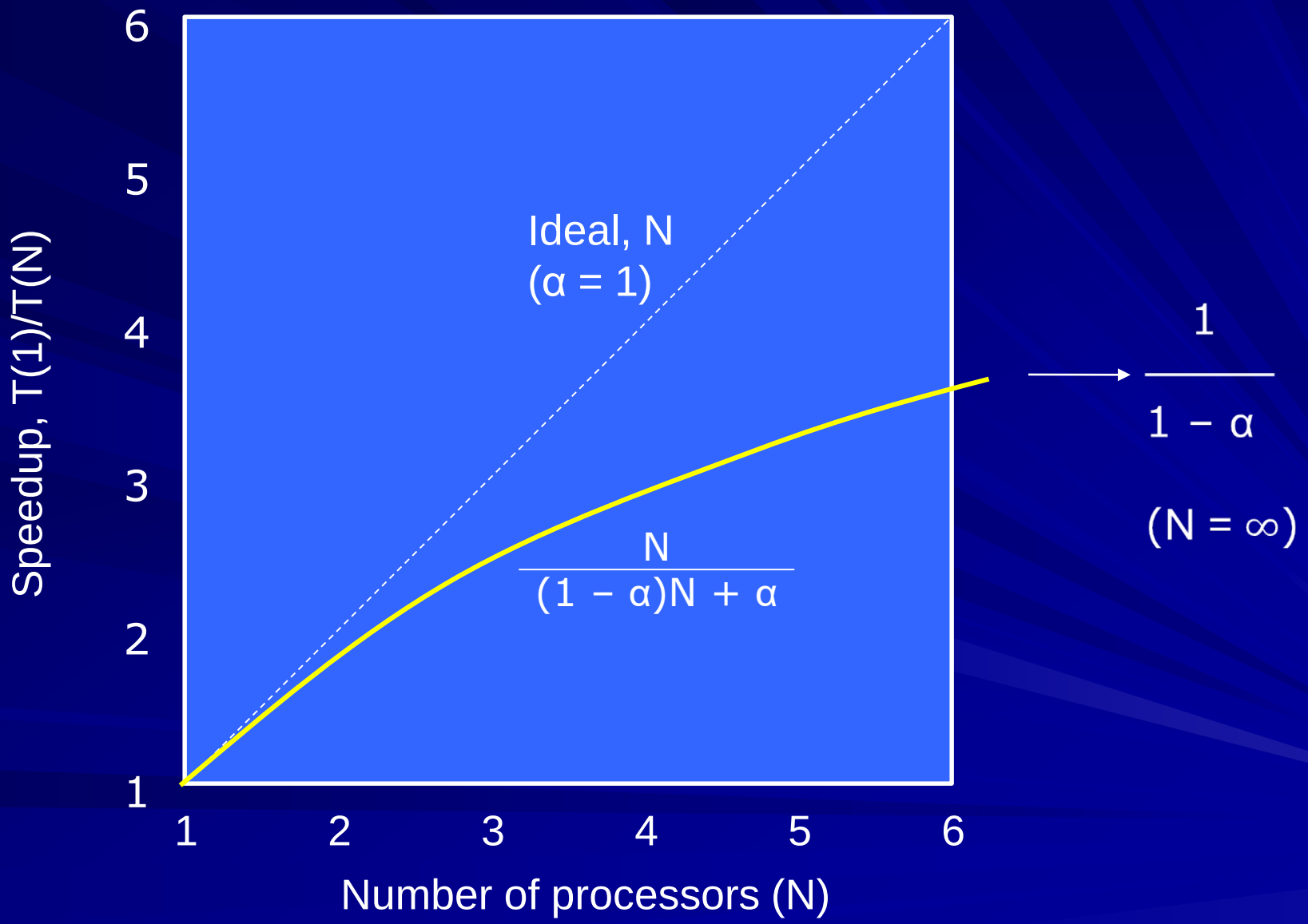
- Speedup $= \dfrac{T(1)}{T(N)} = \dfrac{1}{1 - \alpha + \alpha/N} = \dfrac{N}{(1 - \alpha)N + \alpha}$

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

Run Time


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

Speedup


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

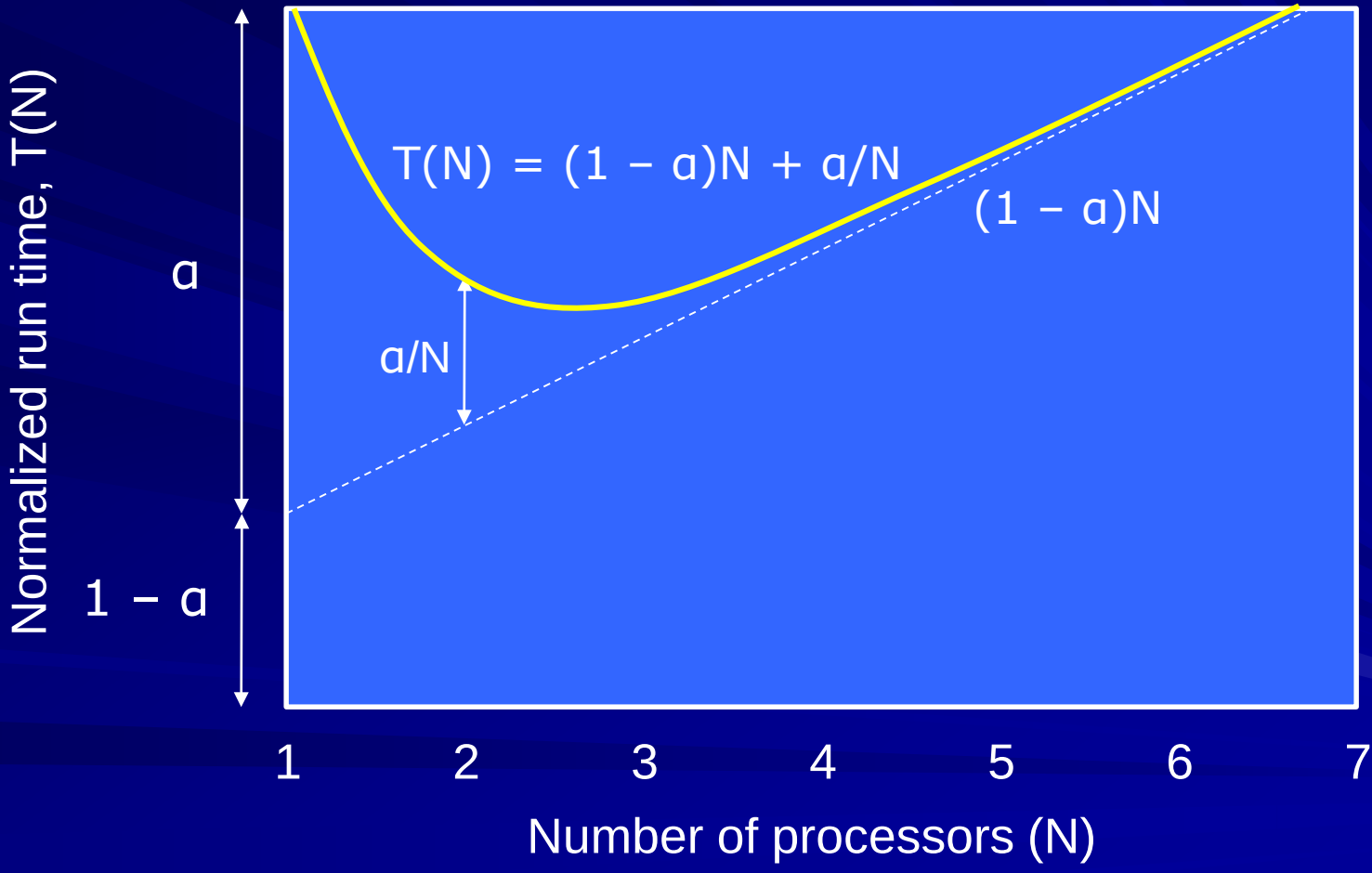
Example

- 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


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

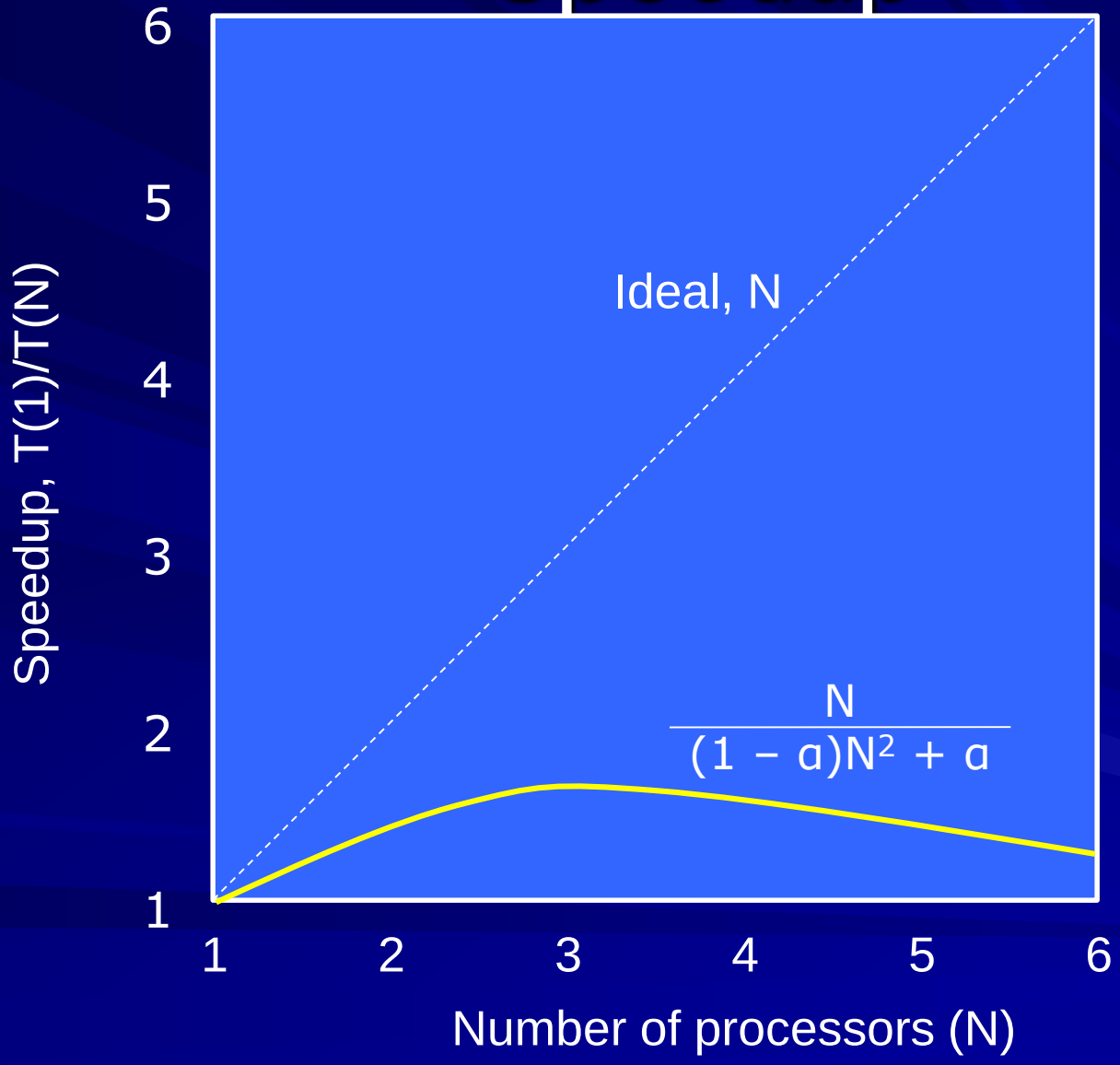
Minimum Run Time

- Minimize N processor run time,

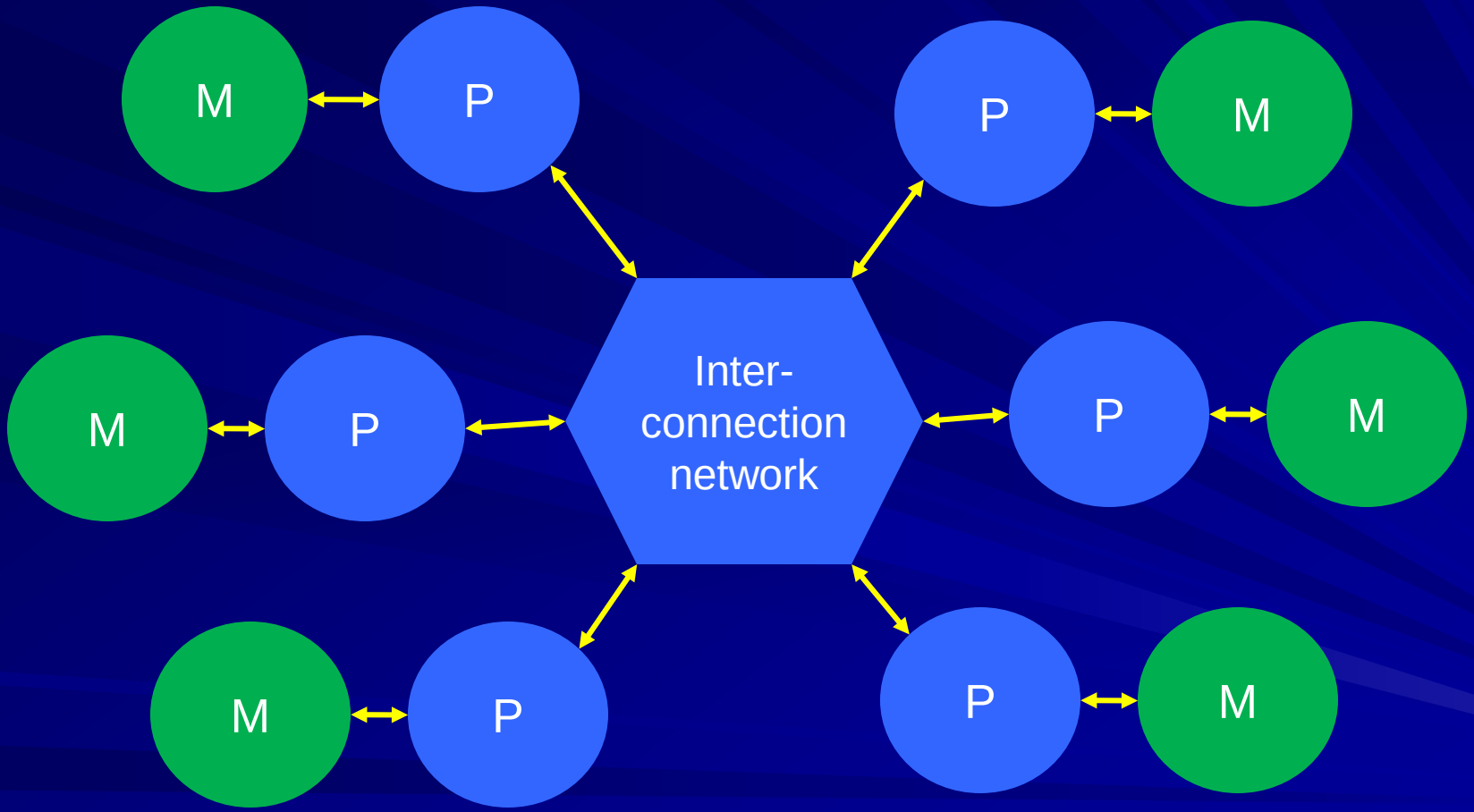
$$T(N) = (1 - \alpha)N + \alpha/N$$

- $\partial T(N)/\partial N = 0$
- $1 - \alpha - \alpha/N^2 = 0$, $N = [\alpha/(1 - \alpha)]^{1/2}$
- Min. $T(N) = 2[\alpha(1 - \alpha)]^{1/2}$, because $\partial^2 T(N)/\partial N^2 > 0$.
- Maximum speedup = $1/T(N) = 0.5[\alpha(1 - \alpha)]^{-1/2}$
- 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
N
(1 – α)N² + α
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 = $1 - \alpha$, same as single processor
- Communication overhead = $\beta(N - 1)$
- N processor run time, $T(N) = \beta(N - 1) + 1/N$
- $$\text{Speedup} = \frac{1}{\beta(N - 1) + 1/N} = \frac{N}{\beta N(N - 1) + 1}$$

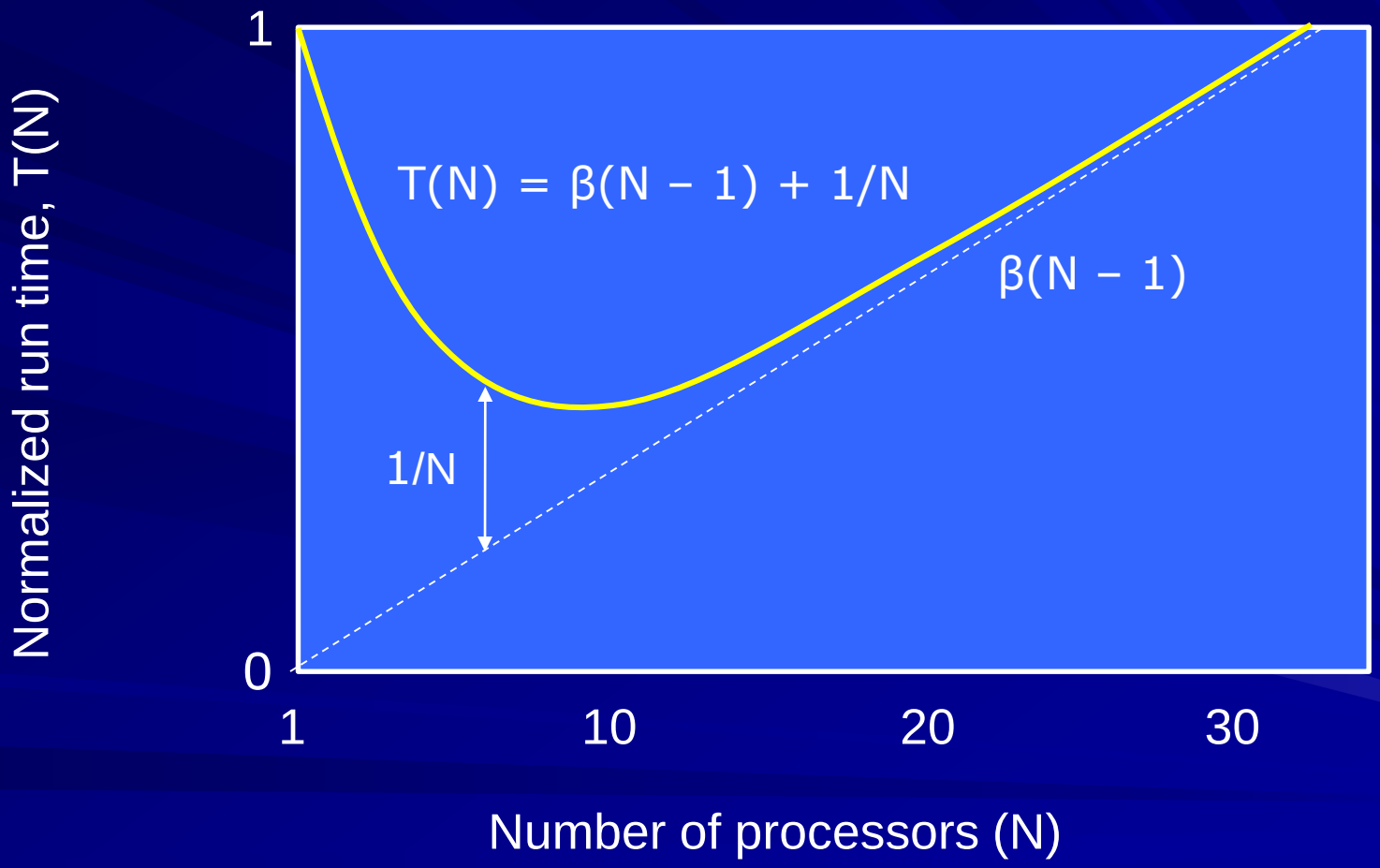
Minimum Run Time

- Minimize N processor run time,

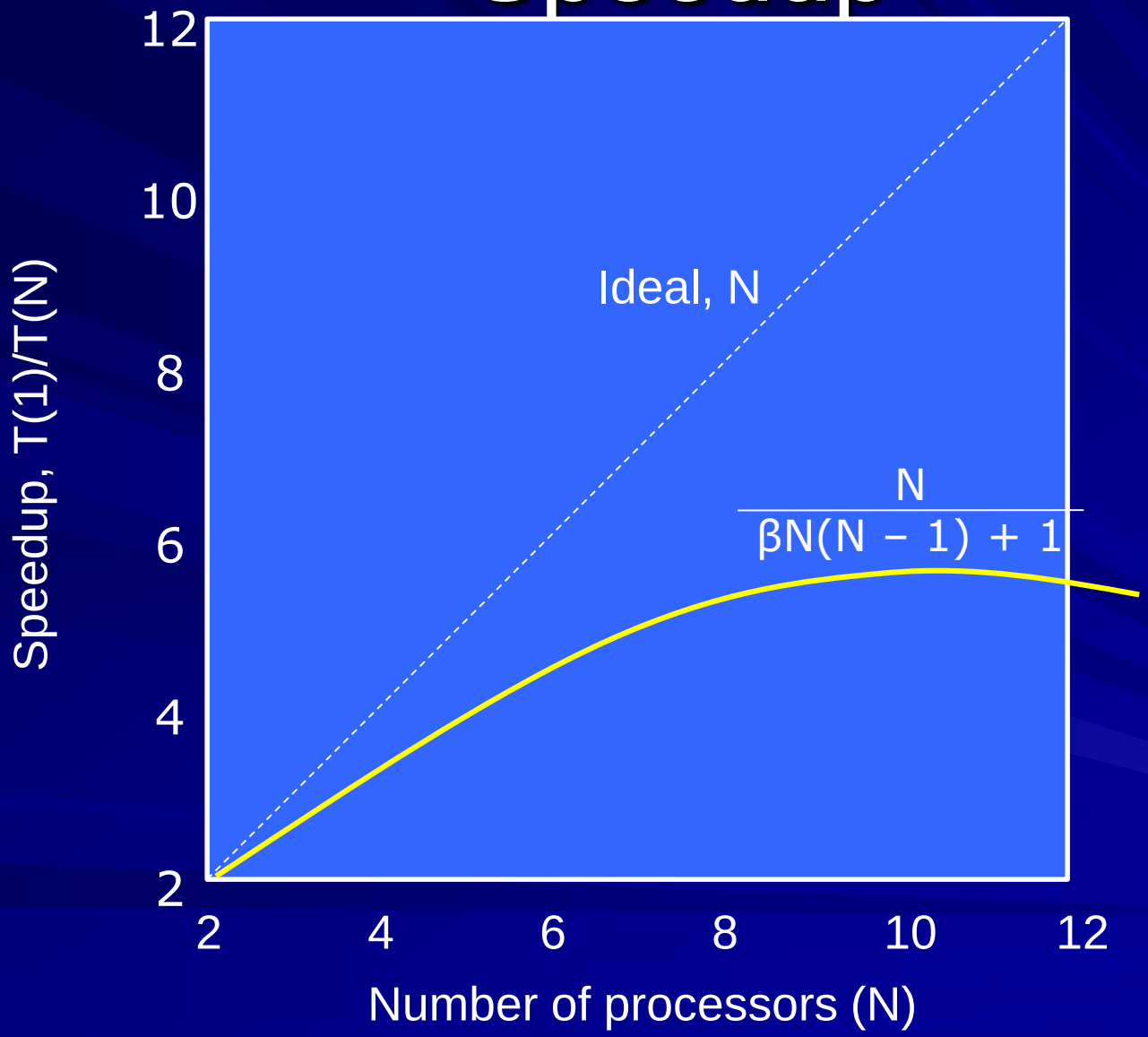
$$T(N) = \beta(N - 1) + 1/N$$

- $\partial T(N)/\partial N = 0$
- $\beta - 1/N^2 = 0$, $N = \beta^{-1/2}$
- 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
1
10
20
30
T(N) = β(N – 1) + 1/N
β(N – 1)
1/N
Normalized run time, T(N)
Number of processors (N)

Speedup


12
10
8
6
4
2
Speedup, T(1)/T(N)
Ideal, N
N
βN(N − 1) + 1
2
4
6
8
10
12
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.
- U. Vishkin, “Is Multicore Hardware for General-Purpose Parallel Processing Broken?” *Comm. ACM*, vol. 57, no. 4, pp. 35-39, Apr. 2014.