CSC 411

Computer Organization (Fall 2024)
Lecture 16: Performance

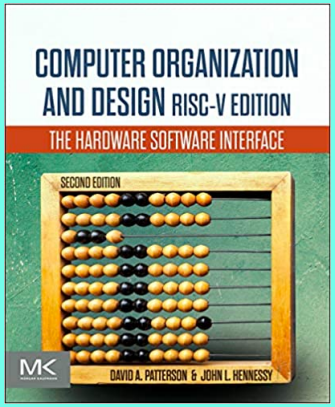
Prof. Marco Alvarez, University of Rhode Island

Disclaimer

Some figures and slides are adapted from:

Computer Organization and Design (Patterson and Hennessy)

The Hardware/Software Interface



Defining performance

- Performance
  - how efficiently a system can perform a task
  - a multi-faceted concept that can vary depending on the context and the specific goals considered

Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers × m.p.h.)
Boeing 737	240	3000	564	135,360
BAC/Sud Concorde	132	4000	1350	178,200
Boeing 777-200LR	301	9395	554	166,761
Airbus A380-800	853	8477	587	500,711

- In computing, what factors can impact performance?
  - algorithm, determines number of operations executed
  - language/compiler/architecture determines number of machine instructions executed per operation
  - processor/memory determine how fast instructions are executed
  - input/output/OS determine determines how fast I/O operations are executed

Response time and throughput

- **Latency** (response time)
  - total time required to complete a task
  - usually includes I/O, memory latency, system overhead, CPU time, etc.
- **Throughput** (bandwidth)
  - number of tasks completed per unit time
  - e.g., tasks/transactions/… per hour
  - critical in server environments

Performance equations

- Performance
  - higher perfomance (P) => lower execution time (T)

$$P = \frac{1}{T}$$

- **Speedup** (relative performance)
  - measures the performance improvement by one system (X) compared to another baseline or reference system (base)

$$S = \frac{P_X}{P_{base}} = \frac{T_{base}}{T_X}$$

"X is S times faster than baseline"

Practice

- The times taken to run a program on processors A and B are 10s on A, 15s on B
  - what is the speedup of A over B? or, how much faster is A w.r.t. B?

Measuring execution time

- **Elapsed time** (wall clock time, response time)
  - total response time, including all aspects: processing, I/O, OS overhead, idle time
- **CPU execution time** (CPU time)
  - time spent processing a task/job, it discounts I/O time and running other jobs
  - can be divided into **user CPU time** and **system CPU time** (difficult to separate accurately)
- Different applications are affected by different performance aspects
  - one must have a clear definition of what performance metric to use to improve system/program performance

The time command

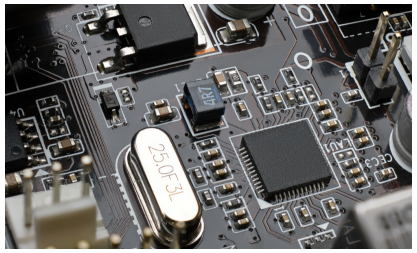
```
$ time (seq 100 | awk '{sum+=$1} END {print sum}')
5050

real    0m0.002s
user    0m0.002s
sys     0m0.002s


$ time (seq 10000000 | awk '{sum+=$1} END {print sum}')
50000005000000

real    0m2.361s
user    0m2.419s
sys     0m0.057s


$ time curl www.godbold.org >/dev/null

real    0m0.209s
user    0m0.004s
sys     0m0.003s
```

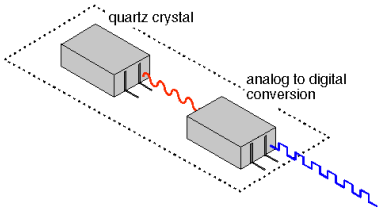
real: wall clock time or total elapsed time including time used by other processes and I/O
user: CPU time spent in user-mode, including only actual CPU time used by the process
sys: CPU time spent in kernel-mode (system calls), including only actual CPU time used by the process

(*) In multicore systems, the **user** and **sys** times may correspond to the sum of each core's CPU times

CPU clocking

- Operation of digital hardware governed by a constant-rate **clock** alternating high-low voltages
- Clock generator
  - a CPU clock crystal is a critical component in modern computer systems that provides a stable and precise timing reference
  - the clock crystal vibrates at a specific frequency when an electric current is applied
  - this vibration is used to produce a continuous and stable **clock signal**
    - signal is used to synchronize and coordinate various operations within the CPU and other system components

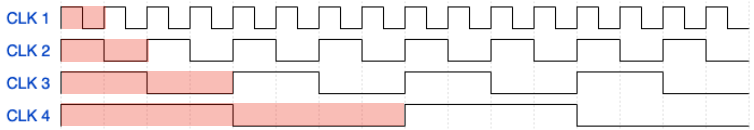




CPU clocking

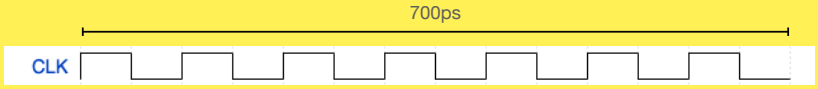
- **Clock frequency (F)** (clock rate)
  - number of clock cycles that occur per second (typically measured in hertz (Hz))
- **Clock period (T)**
  - the amount of time it takes for one complete clock cycle to occur (typically measured in nanoseconds (ns))
  - longer clock period means a slower clock rate, while a shorter clock period indicates a faster clock rate

$$F = \frac{1}{T}$$



Practice

- Identify
  - rising and falling edges
  - duration of clock cycle
  - frequency



Practice

- If the clock period is 250ps, what is the clock frequency?
  - $250ps = 250 \times 10^{-12}s$

$$F = \frac{1}{T} = \frac{1}{250 \times 10^{-12}}$$
$$= \frac{1000 \times 10^{-3}}{250 \times 10^{-12}}$$
$$= 4 \times 10^{9} Hz = 4GHz$$

SI prefixes

Factor	Name	Symbol	Factor	Name	Symbol
10^{1}	deca	da	10^{-1}	deci	d
10^{2}	hecto	h	10^{-2}	centi	c
10^{3}	kilo	k	10^{-3}	milli	m
10^{6}	mega	M	10^{-6}	micro	μ
10^{9}	giga	G	10^{-9}	nano	n
10^{12}	tera	T	10^{-12}	pico	p
10^{15}	peta	P	10^{-15}	femto	f
10^{18}	exa	E	10^{-18}	atto	a
10^{21}	zetta	Z	10^{-21}	zepto	z
10^{24}	yotta	Y	10^{-24}	yocto	y
10^{27}	ronna	R	10^{-27}	ronto	r
10^{30}	quetta	Q	10^{-30}	quecto	q

SI prefixes are a set of 24 prefixes used in the International System of Units (SI) to indicate multiples and submultiples of SI units. They are based on powers of 10, and each prefix has a unique symbol.

Practice

- What is the clock cycle time of a 3 GHz processor?

CPU execution time

$$\text{CPU time} = \text{clock cycles} \times \text{clock cycle time}$$
$$= \frac{\text{clock cycles}}{\text{clock rate}}$$

- Can improve performance by:
  - reducing the number of clock cycles
  - increasing the clock rate (frequency)
  - hardware designers must often trade off clock rate against clock cycles
    - pushing clock frequencies higher has become increasingly difficult and power-hungry
    - reducing cycles requires more transistors for performance enhancement logic

Instruction execution performance

- Instruction count (IC)
  - number of instructions that a processor executes for a specific program
  - IC determined by program, ISA, and compiler
- Average cycles per instruction (CPI)
  - average of clock cycles each instruction takes to execute, over all the instructions executed by the program
  - CPI is determined by the CPU hardware

$$\text{clock cycles} = \text{IC} \times \text{CPI}$$
$$\text{CPU time} = \text{IC} \times \text{CPI} \times \text{clock cycle time}$$
$$= \frac{\text{IC} \times \text{CPI}}{\text{clock rate}}$$

Practice

- Computer A: clock cycle time = 250ps, CPI = 2.0
- Computer B: clock cycle time = 500ps, CPI = 1.2
  - if running the same instructions, assuming same ISA, which computer is faster, and what is the speedup?

$$\begin{aligned}\text{CPU time}_A &= \text{IC} \times \text{CPI}_A \times \text{clock cycle time}_A \\ &= \text{IC} \times 2.0 \times 250ps = \text{IC} \times 500ps \\ \text{CPU time}_B &= \text{IC} \times \text{CPI}_B \times \text{clock cycle time}_B \\ &= \text{IC} \times 1.2 \times 500ps = \text{IC} \times 600ps \\ \frac{\text{CPU time}_B}{\text{CPU time}_A} &= \frac{\text{IC} \times 600ps}{\text{IC} \times 500ps} = 1.2\end{aligned}$$

Computer A is 1.2x faster than Computer B

Practice

- Consider 3 different processors executing the same program (same instructions)
  - P1 has a clock cycle time of 0.33ns and a CPI of 1.5
  - P2 has a clock cycle time of 0.40ns and a CPI of 1
  - P3 has a clock cycle time of 0.25ns and a CPI of 2.2
- Questions
  - which processor has the highest clock rate? what is it?
  - which processor is the fastest executing the program?
  - which is the slowest?

CPI in more detail

- If different instruction classes take different numbers of cycles

$$\text{clock cycles} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \text{IC}_i\right)$$

CPI for group i

instruction count for group i

- Calculating the (average) CPI using a weighted average

n classes

$$\text{CPI} = \frac{\sum_{i=1}^{n} \left(\text{CPI}_i \times \text{IC}_i\right)}{\text{IC}}$$

Practice

- Assume two programs using instructions in classes A, B, C.
  - Which program executes the most instructions? Which program is faster (clock cycles)? What is the CPI for each sequence?

Class	A	B	C
CPI for class	5	10	4
IC in program 1	100	250	250
IC in program 2	300	150	100

- Program 1
  - IC =
  - clock cycles =
  - CPI =
- Program 2
  - IC =
  - clock cycles =
  - CPI =

Key takeaways

- Factors influencing performance

$$\text{CPU time} = \text{IC} \times \text{CPI} \times \text{clock cycle time}$$

- Performance depends on:
  - **algorithm**: affects IC, possibly CPI
  - **programming language**: affects IC, CPI
  - **compiler**: affects IC, CPI
  - **ISA**: affects IC, CPI, clock cycle time

```
MIPS (Millions of Instructions Per Second) can be a misleading metric for
comparing performance. Different processors have different ISAs and not all
    instructions are equal. A high MIPS might indicate high instruction
throughput, but it doesn't necessarily translate to real-world performance.
    It's often better to rely on benchmarks that test specific workloads.
```

Practice

- Architecture 1: (RISC)
  - a program runs 4 billion instructions
  - each instruction completes in an average of 1.5 cycles (CPI)
  - clock speed is 1 GHz
- Architecture 2: (CISC)
  - same program runs 2 billion instructions
  - each instruction completes in an average of 6 cycles (CPI)
  - clock speed is 1.5 GHz
- Which CPU time is better?

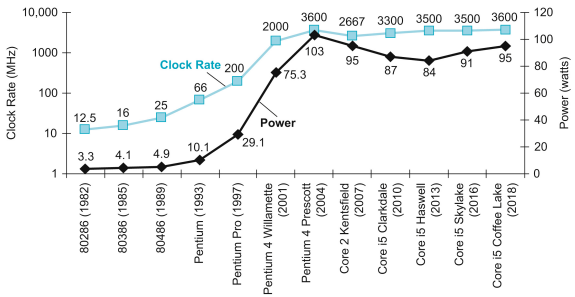
Practice

- Consider two processors P1 and P2, implementing the same ISA
  - the instructions can be grouped into four classes (A, B, C, D), with the following CPIs:
    - processor P1 with a clock rate of 2.5GHz and CPIs 1, 2, 2, 4
    - processor P2 with a clock rate of 3GHz and CPIs 2, 2, 1, 1
  - given a program with 10^6 instructions split as: 20% class A, 20% class B, 40% class C, and 20% class D
    - which processor runs the program faster P1 or P2?
    - find the number of clock cycles required in both cases

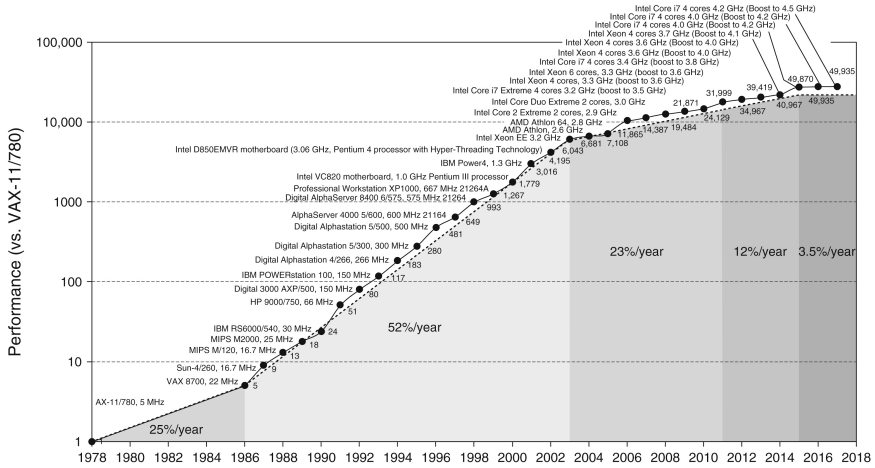
Multicore performance

Power trends

- Clock rate and power are correlated
  - Power $\propto 1/2 \times$ capacitive load $\times$ voltage$^2 \times$ frequency
  - both voltage and frequency significantly impact power, which directly translates to heat generation



(uni) Processor performance

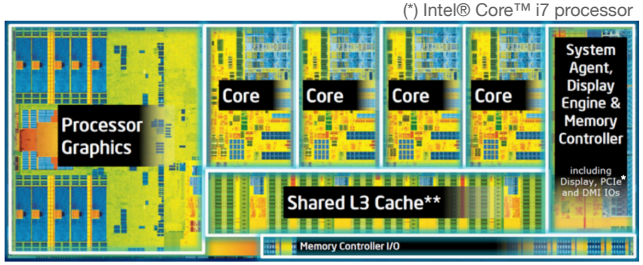


(*) Performance relative to the VAX 11/780 as measured by the SPECint benchmarks

Multicore

- Multicore microprocessors
  - more than one processor per chip

(*) Intel® Core™ i7 processor

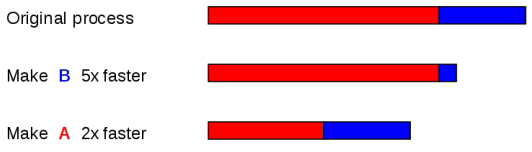


- Requires explicit parallel programming
  - challenges: programming for performance, load balancing, optimizing communication and synchronization

Multicore

processors	exec. time/ processor	time w/overhead	speedup	actual speedup/ideal speedup
1	100			
2	50	54	100/54 = 1.85	1.85/2 = .93
4	25	29	100/29 = 3.44	3.44/4 = 0.86
8	12.5	16.5	100/16.5 = 6.06	6.06/8 = 0.75
16	6.25	10.25	100/10.25 = 9.76	9.76/16 = 0.61

- What is happening?



https://en.wikipedia.org/wiki/Amdahl's_law

Amdahl's law

- "the overall performance improvement gained by optimizing a single part of a system is limited by the fraction of time that the improved part is actually used"
  - improving an aspect of a computer does not translate automatically into a proportional improvement in overall performance
- Architecture design is bottleneck-driven
  - make the common case fast
  - do not waste resources on a component that has little impact on overall performance/power

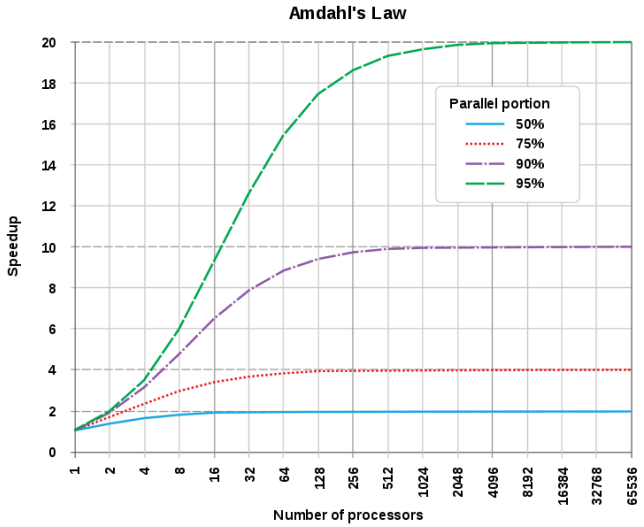
Amdahl's law

- Assume:
  - S: overall system speedup, F: fraction enhanced, I: factor of improvement

$$S = \frac{1}{\underbrace{(1 - F)}_{\text{non-enhanced}} + \underbrace{\frac{F}{I}}_{\text{enhanced}}}$$

- Example
  - 40% of instructions in program A are multiplications, and a new processor runs multiplications faster by a factor of 3.4
  - What is the overall system speedup?

Amdahl's law and parallel computers



https://en.wikipedia.org/wiki/Amdahl's_law

Practice

- A program spends 40% of time in the CPU and 60% doing I/O
  - a new processor **10x faster** results in a reduction in total execution time
  - what is the speedup?
  - according to Amdahl's Law, by continuously increasing the processor's performance, the **maximum speedup** will be 1.66

Comparing performance

Summarizing performance

- Quantifying and comparing performance is crucial in computer systems
  - these measures are essential for benchmarking, performance analysis, and making informed architectural decisions
- Two commonly used measures to summarize a set of performance values into a single value
  - the **arithmetic mean**
  - the **geometric mean**

Summarizing performance

- The arithmetic mean
  - summing up the individual performance values and dividing by the number of values
  - suitable for **absolute values** like CPU clock cycles, execution times, etc.

$$\frac{1}{n}\sum_{i=1}^{n} P_i$$

- The geometric mean
  - multiplying the individual performance values and taking the Nth root, where N is the number of values
  - suitable for **percentages or ratios** (relative performance) like speedups, cache hit rates, etc.

$$\sqrt[n]{\prod_{i=1}^{n} P_i}$$

Measuring performance

SPEC

- Standards Performance Evaluation Corporation
  - non-profit corporation formed in 1988 to establish, maintain and endorse standardized benchmarks and tools to evaluate performance and energy efficiency for the newest generation of computing systems
  - membership is comprised of more than 120 computer hw/sw vendors, educational institutions, research organizations, and government agencies worldwide.
  - https://www.spec.org/
- Groups:
  - Research Group (RG)
  - Open Systems Group (OSG)
    - Cloud, CPU, Java, Power, Virtual Machine, File Server
  - High Performance Group (HPG)
    - MPI, OpenMP, OpenCL, OpenACC
  - Graphics and Workstation Performance Group (GWPG)

SPEC benchmarks

- Benchmark suites
  - standardized workloads designed to mimic real-world applications
  - measure performance across different aspects of a system: CPU, memory, I/O
- Different benchmark suites for various computing needs
  - CPU, graphics, web applications, HPC, etc.
- The result of a SPEC benchmark suite is always a **SPEC score**
  - higher is better and score is always **relative to a reference machine**
    - each benchmark has its own reference machine
  - some benchmarks have a power score, in addition to a performance score
- Results are published on the SPEC's website for comparison

SPEC CPU

Q6. What does SPEC CPU 2017 measure?

SPEC CPU 2017 focuses on compute intensive performance, which means these benchmarks emphasize the performance of:

- **Processor** - The CPU chip(s).
- **Memory** - The memory hierarchy, including caches and main memory.
- **Compilers** - C, C++, and Fortran compilers, including optimizers.

SPEC CPU 2017 intentionally depends on all three of the above - not just the processor.

SPEC CPU 2017 is not intended to stress other computer components such as networking, graphics, Java libraries, or the I/O system. Note that there are other SPEC benchmarks that focus on those areas.

Q8. What does SPEC provide?

SPEC CPU 2017 is distributed as an ISO image that contains:

- Source code for the benchmarks
- Data sets
- A tool set for compiling, running, validating and reporting on the benchmarks
- Pre-compiled tools for a variety of operating systems
- Source code for the SPEC CPU 2017 tools, for systems not covered by the pre-compiled tools
- Documentation
- Run and reporting rules

The documentation is also available at www.spec.org/cpu2017/Docs/index.html , including the Unix and Windows installation guides.

Q12. What is a CPU 2017 "suite"?

A *suite* is a set of benchmarks that are run as a group to produce one of the overall metrics.

SPEC CPU 2017 includes four suites that focus on different types of compute intensive performance:

Short Tag	Suite	Contents	Metrics	How many copies? What do Higher Scores Mean?
intspeed	SPECspeed® 2017 Integer	10 integer benchmarks	SPECspeed®2017_int_base SPECspeed®2017_int_peak SPECspeed®2017_int_energy_base SPECspeed®2017_int_energy_peak	SPECspeed suites always run one copy of each benchmark. Higher scores indicate that less time is needed.
fpspeed	SPECspeed®2017 Floating Point	10 floating point benchmarks	SPECspeed®2017_fp_base SPECspeed®2017_fp_peak SPECspeed®2017_fp_energy_base SPECspeed®2017_fp_energy_peak	
intrate	SPECrate® 2017 Integer	10 integer benchmarks	SPECrate®2017_int_base SPECrate®2017_int_peak SPECrate®2017_int_energy_base SPECrate®2017_int_energy_peak	SPECrate suites run multiple concurrent copies of each benchmark. The tester selects how many. Higher scores indicate more *throughput* (work per unit of time).
fprate	SPECrate® 2017 Floating Point	13 floating point benchmarks	SPECrate®2017_fp_base SPECrate®2017_fp_peak SPECrate®2017_fp_energy_base SPECrate®2017_fp_energy_peak	

Q13. What are the benchmarks?

SPEC CPU 2017 has 43 benchmarks, organized into 4 suites:

SPECrate®2017 Integer	SPECspeed®2017 Integer	Language [1]	KLOC [2]	Application Area
500.perlbench_r	600.perlbench_s	C	362	Perl interpreter
502.gcc_r	602.gcc_s	C	1,304	GNU C compiler
505.mcf_r	605.mcf_s	C	3	Route planning
520.omnetpp_r	620.omnetpp_s	C++	134	Discrete Event simulation - computer network
523.xalancbmk_r	623.xalancbmk_s	C++	520	XML to HTML conversion via XSLT
525.x264_r	625.x264_s	C	96	Video compression
531.deepsjeng_r	631.deepsjeng_s	C++	10	Artificial Intelligence: alpha-beta tree search (Chess)
541.leela_r	641.leela_s	C++	21	Artificial Intelligence: Monte Carlo tree search (Go)
548.exchange2_r	648.exchange2_s	Fortran	1	Artificial Intelligence: recursive solution generator (Sudoku)
557.xz_r	657.xz_s	C	33	General data compression

SPECrate®2017 Floating Point	SPECspeed®2017 Floating Point	Language [1]	KLOC [2]	Application Area
503.bwaves_r	603.bwaves_s	Fortran	1	Explosion modeling
507.cactuBSSN_r	607.cactuBSSN_s	C++, C, Fortran	257	Physics: relativity
508.namd_r		C++	8	Molecular dynamics
510.parest_r		C++	427	Biomedical imaging: optical tomography with finite elements
511.povray_r		C++, C	170	Ray tracing
519.lbm_r	619.lbm_s	C	1	Fluid dynamics
521.wrf_r	621.wrf_s	Fortran, C	991	Weather forecasting
526.blender_r		C++, C	1,577	3D rendering and animation
527.cam4_r	627.cam4_s	Fortran, C	407	Atmosphere modeling
	628.pop2_s	Fortran, C	338	Wide-scale ocean modeling (climate level)
538.imagick_r	638.imagick_s	C	259	Image manipulation
544.nab_r	644.nab_s	C	24	Molecular dynamics
549.fotonik3d_r	649.fotonik3d_s	Fortran	14	Computational Electromagnetics
554.roms_r	654.roms_s	Fortran	210	Regional ocean modeling

[1] For multi-language benchmarks, the first one listed determines library and link options (details)

[2] KLOC = line count (including comments/whitespace) for source files used in a build / 1000

Q15. What are "SPECspeed" and "SPECrate" metrics?

There are many ways to measure computer performance. Among the most common are:

- Time - For example, seconds to complete a workload.
- Throughput - Work completed per unit of time, for example, jobs per hour.

SPECspeed is a time-based metric; SPECrate is a throughput metric.

Calculating SPECspeed® Metrics	Calculating SPECrate® Metrics
1 copy of each benchmark in a suite is run.	The tester chooses how many concurrent copies to run
The tester may choose how many OpenMP threads to use.	OpenMP is disabled.
For each benchmark, a performance ratio is calculated as: `time on a reference machine / time on the SUT`	For each benchmark, a performance ratio is calculated as: `number of copies * (time on a reference machine / time on the SUT)`
Higher scores mean that less time is needed.	Higher scores mean that more work is done per unit of time.
Example: • The reference machine ran 600.perlbench_s in 1775 seconds. • A particular SUT took about 1/5 the time, scoring about 5. • More precisely: `1775/354.329738 = 5.009458`	Example: • The reference machine ran 1 copy of 500.perlbench_r in 1592 seconds. • A particular SUT ran 8 copies in about 1/3 the time, scoring about 24. • More precisely: `8*(1592/541.52471) = 23.518776`

For both SPECspeed and SPECrate, in order to provide some assurance that results are repeatable, the entire process is repeated.
The tester may choose:

a. To run the suite of benchmarks three times, in which case the tools select the medians.
b. Or to run twice, in which case the tools select the lower ratios (i.e. slower). *New with CPU 2017*

For both SPECspeed and SPECrate, the selected ratios are averaged using the Geometric Mean, which is reported as the overall metric.

For the energy metrics, the calculations are done the same way, using `energy` instead of `time` in the above formulas.

The reference times and reference energy are based on the observations posted with www.spec.org/cpu2017/results/
1, 2, 3, and 4
The measured values, with all available digits, can be found in the CSV versions:
1, 2, 3, and 4.
The reference times used in the calculations are rounded to the next higher integral number of seconds.

Example

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Perl interpreter	perlbench	2684	0.42	0.556	627	1774	2.83
GNU C compiler	gcc	2322	0.67	0.556	863	3976	4.61
Route planning	mcf	1786	1.22	0.556	1215	4721	3.89
Discrete Event simulation - computer network	omnetpp	1107	0.82	0.556	507	1630	3.21
XML to HTML conversion via XSLT	xalancbmk	1314	0.75	0.556	549	1417	2.58
Video compression	x264	4488	0.32	0.556	813	1763	2.17
Artificial Intelligence: alpha-beta tree search (Chess)	deepsjeng	2216	0.57	0.556	698	1432	2.05
Artificial Intelligence: Monte Carlo tree search (Go)	leela	2236	0.79	0.556	987	1703	1.73
Artificial Intelligence: recursive solution generator (Sudoku)	exchange2	6683	0.46	0.556	1718	2939	1.71
General data compression	xz	8533	1.32	0.556	6290	6182	0.98
Geometric mean	–	–	–	–	–	–	2.36

Running on a 1.8 GHz Intel Xeon E5-2650L