2.2 Performance metrics

Dec — Binary

10^3	2^{10}	1 Ki	1 KiB one kilobyte
10^6	2^{20}	1 Mi	
10^9	2^{30}	1 GiB	
10^{12}	2^{40}	1 TiB	
10^{15}	2^{50}	1 PiB	peta
10^{18}	2^{60}	1 EiB	exa
10^{21}	2^{70}	1 ZiB	zetta
10^{24}	2^{80}	1 YiB	yotta

$$\text{Performance}_x = \frac{1}{\text{Execution time}_x}$$

- Real programs (true)
- Not → toy programs
  - synthetic benchmarks
  - kernels

- transitivity $A > B$ & $B > C$ => $A > C$

Benchmark suites

SPEC - Standard Performance Evaluation Corporation www.spec.org

1989, 92, 95, 2000, 2006, 2017

- CPU SPEC
- SPEC graphics
- SPEC Java

Benchmark name by SPEC generation

SPEC2006 benchmark description	SPEC2006	SPEC2000	SPEC95	SPEC92	SPEC89
GNU C compiler					gcc
Interpreted string processing			perl		espresso
Combinatorial optimization		mcf			li
Block-sorting compression		bzip2		compress	eqntott
Go game (AI)	go	vortex	go	sc	
Video compression	h264avc	gzip	ijpeg		
Games/path finding	astar	eon	m88ksim		
Search gene sequence	hmmer	twolf			
Quantum computer simulation	libquantum	vortex			
Discrete event simulation library	omnetpp	vpr			
Chess game (AI)	sjeng	crafty			
XML parsing	xalancbmk	parser			
CFD/blast waves	bwaves				fpppp
Numerical relativity	cactusADM				tomcatv
Finite element code	calculix				doduc
Differential equation solver framework	dealII				nasa7
Quantum chemistry	gamess				spice
EM solver (freq/time domain)	GemsFDTD			swim	matrix300
Scalable molecular dynamics (~NAMD)	gromacs		apsi	hydro2d	
Lattice Boltzman method (fluid/air flow)	lbm		mgrid	su2cor	
Large eddie simulation/turbulent CFD	LESlie3d	wupwise	applu	wave5	
Lattice quantum chromodynamics	milc	apply	turb3d		
Molecular dynamics	namd	galgel			
Image ray tracing	povray	mesa			
Spare linear algebra	soplex	art			
Speech recognition	sphinx3	equake			
Quantum chemistry/object oriented	tonto	facerec			
Weather research and forecasting	wrf	ammp			
Magneto hydrodynamics (astrophysics)	zeusmp	lucas			
		fma3d			
		sixtrack			

Server benchmarks

- Throughput-centric metric
- Correct CPU time data arate
- mix CPU SPEC suites

SPEC

2.2.1 Reporting performance results

SPEC Ratio

$$\frac{\text{SPEC Ratio}_A}{\text{SPEC Ratio}_B} = \frac{\frac{\text{Execution time reference}}{\text{Execution time}_A}}{\frac{\text{Execution time reference}}{\text{Execution time}_B}} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{\text{Performance}_A}{\text{Performance}_B}$$

$$\text{Geometric mean} = \sqrt[n]{\prod_{i=1}^{n} \text{Sample}_i}, \quad \text{Sample}_i \rightarrow \text{SPEC}_{\text{ratio}_i} \rightarrow \text{program}$$

① The geometric means of ratios = the ratio of geometric means

② The ratio of geometric means = the geometric mean of performance ratios

$$\frac{\text{Geometric mean}_A}{\text{Geometric mean}_B} = \frac{\sqrt[n]{\prod_{i=1}^{n} \text{SPECratio}_{Ai}}}{\sqrt[n]{\prod_{i=1}^{n} \text{SPECratio}_{Bi}}} = \quad (n = 29)$$

$$= \sqrt[n]{\prod_{i=1}^{n} \frac{\text{SPECratio}_{Ai}}{\text{SPECratio}_{Bi}}} = \sqrt[n]{\prod_{i=1}^{n} \frac{\frac{\text{Execution time reference}_i}{\text{Execution time}_{Ai}}}{\frac{\text{Execution time reference}_i}{\text{Execution time}_{Bi}}}} =$$

$$= \sqrt[n]{\prod_{i=1}^{n} \frac{\text{Execution time}_{Bi}}{\text{Execution time}_{Ai}}} = \sqrt[n]{\prod_{i=1}^{n} \frac{\text{Performance}_{Ai}}{\text{Performance}_{Bi}}}$$

$$\text{stder} = \sqrt{\sum_{i=1}^{n} (\text{Sample}_i - \text{Mean})^2}, \quad \text{Sample}_i \rightarrow \text{SPECRatio}_i$$

Geometric standard deviation

$$\text{Geometric mean} = \exp\left(\frac{1}{n} \times \sum_{i=1}^{n} \ln(\text{Sample}_i)\right)$$

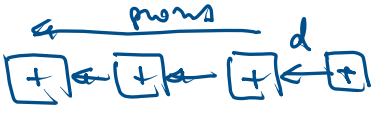
$$\text{gstdev} = \exp\left(\sqrt{\frac{1}{n} \times \sum_{i=1}^{n} \left[\ln(\text{Sample}_i) - \ln(\text{Geometric mean})\right]^2}\right)$$

2.3 Quantitative principles of computer design

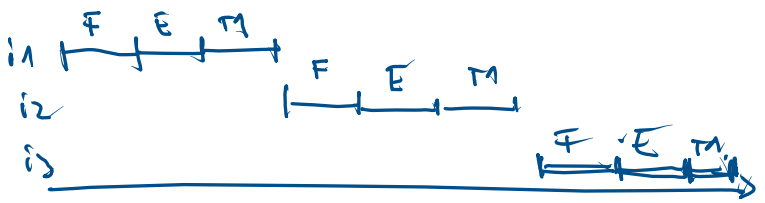
SLD – System-Level Design

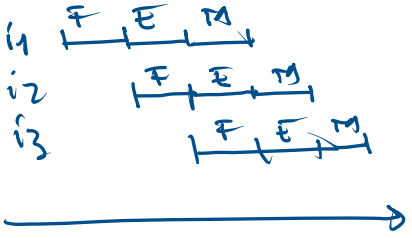
① Take advantage of parallelism (when possible)

CLA



Instruction-level parallelism – pipelining





② Principle of locality

Obs Programs tend to reuse data and instructions they have used recently

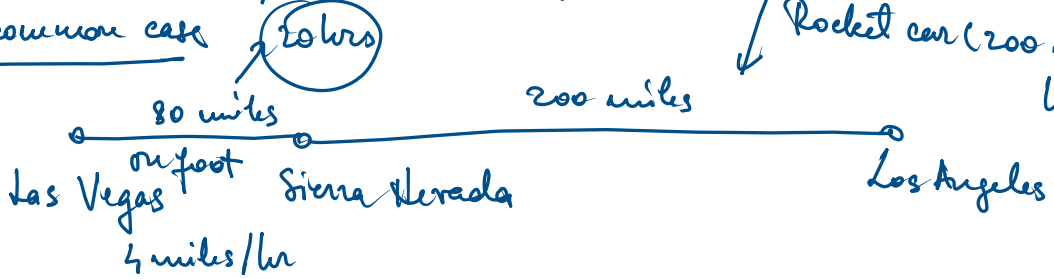
Rule of thumb

A program spends 90% of the time on 10% of the code

- Mostly applied on instructions
- Also works on data
- 2 types of locality → temporal locality, spatial locality

③ Focus on the common case

Amdahl's law



Vehicle for the 2nd part	Hrs for the 2nd part	Speedup in the desert	Hrs for the entire trip	Speedup entire trip
Feet	50	1	70	1
Bike	20	2.5	40	1.75
Hyundai	4	12.5	24	2.9
Ferrari	1.67	30	21.67	3.23
Rocket car	1	50	21	3.3

$$\text{Speedup entire system} = \frac{\text{Performance of the tasks using the speedup when possible}}{\text{Performance of the task without the speedup}} =$$

$$= \frac{\text{Execution time of task without speedup}}{\text{Execution time of task using speedup when possible}} =$$

$$= \frac{\text{Execution time of task without speedup}}{\text{Execution time of task without speedup}\left[(1 - \text{Fraction enhanced}) + \frac{\text{Fraction enhanced}}{\text{Speedup}}\right]} =$$

$$= \frac{1}{(1 - \text{Fraction enhanced}) + \frac{\text{Fraction enhanced}}{\text{Speedup}}}$$

$\frac{200}{280}$ — $(1 - \text{Fraction enhanced})$

$\frac{200}{280}$ — $\frac{\text{Fraction enhanced}}{\text{Speedup}}$ → Speedup in the desert

Limit of system enhancement

$$\lim_{\text{speedup} \to \infty} \text{Speedup for the entire system} = \frac{1}{1 - \text{Fraction enhanced}}$$

2.4 Computer Performance equation

CPU time = Clock cycles for a program × Clock cycle time

Clock cycle time = $\frac{1}{\text{Clock rate}}$

↑ sec ↑ Hz

Clock cycles for a program = Instruction Count × Clock cycles per Instruction

IC integer CPI average

10% 10%

CPU time = IC × CPI × Clock cycle time =>

$$\text{CPU time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{seconds}}{\text{clock cycle}} = \frac{\text{seconds}}{\text{Program}}$$

RISC processors

CISC

$$CPI = \frac{\sum_{i=1}^{n} IC_i \times CPI_i}{IC} = \sum_{i=1}^{n} \frac{IC_i}{IC} \times CPI$$

Fraction of instructions of type i

Example 1

Measurements for a computer system

frequency of FP operations = 25%

Average CPI of FP operations = 4.0 clock cycles

Average CPI of all other instructions = 1.33 clock cycles

Frequency of FPSQRT is 2%

CPI of FPSQRT is 20 clock cycles

Assume 2 design alternatives:

A. decrease of CPI of FPSQRT to 2.

B. decrease the average CPI of all FP operations to 2.5

Which alternative is best?