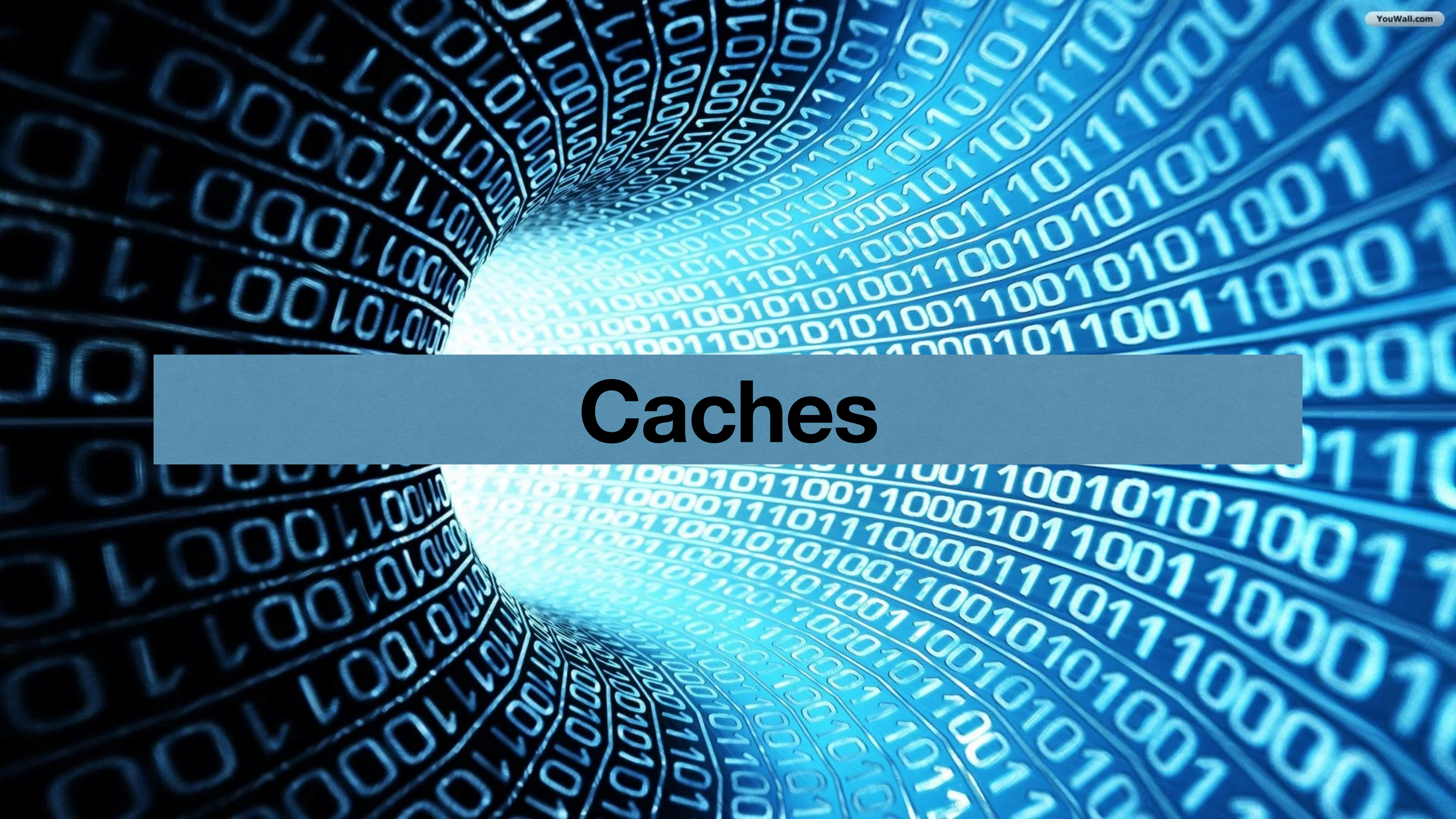
Digital Logic Design + Computer Architecture

Sayandeep Saha

Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology Bombay



Caches

A Few Words About Performance

Performance: Time (Iron Law)

Time/Program =

Instructions/program X cycles/instruction X Time/cycle

Source code	ISA	microarch.
Compiler	microarch.	technology
ISA		

Performance

- *Latency* (execution/response time): time to finish one task. It is additive (Performance = 1/latency)
- *Throughput* (bandwidth): number of tasks/unit time. It is not additive

Performance — In our words

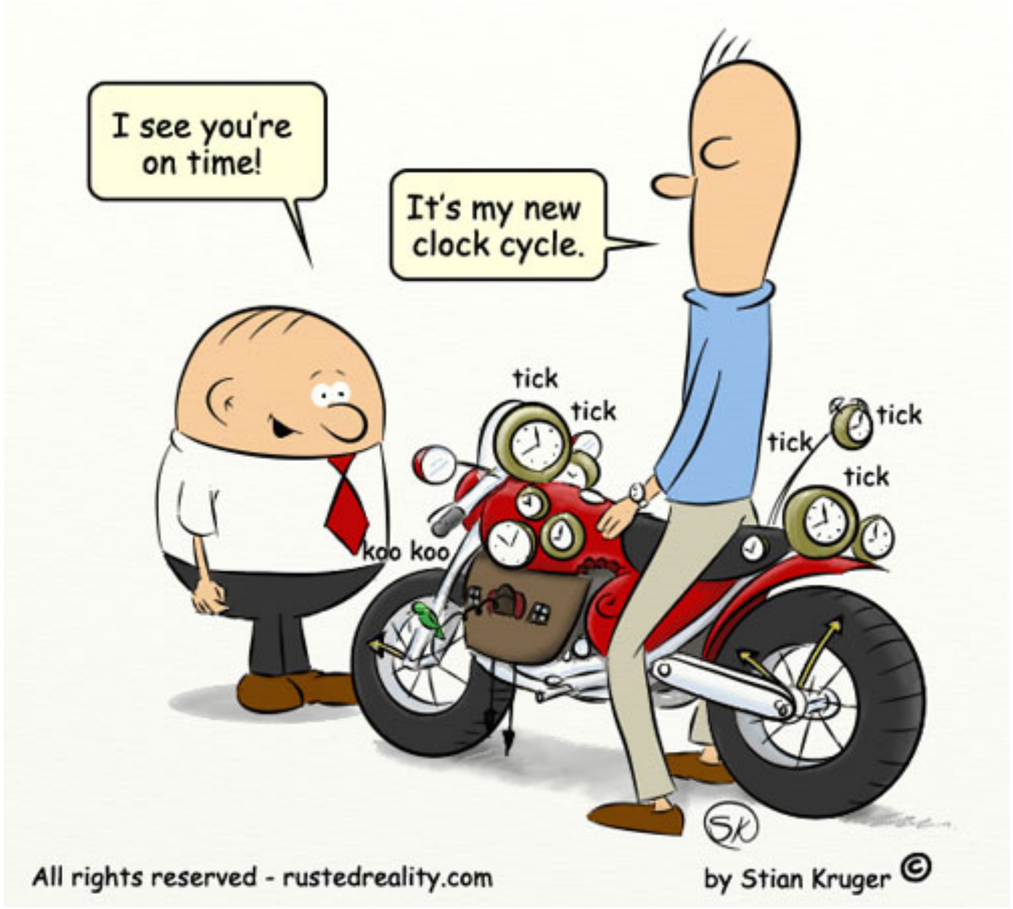
Which computer is faster?

The measure is *execution time*

“X is n times faster than Y”

$$n = \frac{Exetime_Y}{Exetime_X} = \frac{Perfromance_X}{Perfromance_Y}$$

Empirical Evaluation



- Benchmarks
- Metrics
- Simulators
- Latency and bandwidth

Evaluation

- To compare Processor A with Processor B by running programs
- How many programs?
- The programs that you care.
- What if I want to build a new one (processor, caches, DRAM) ?

World of Benchmarks

- SPEC CPU 2017 (https://www.spec.org/cpu2017/)

The **SPEC CPU® 2017** benchmark package contains SPEC's next-generation, industry-standardized, CPU intensive suites for measuring and comparing compute intensive performance, stressing a system's processor, memory subsystem and compiler.

SPECspeed: used for comparing time for a computer to complete single tasks

SPECrate: measure the throughput or work per unit of time.

What are there? From a GCC compiler, Gaming, Video compression, Chess(AI), Differential Equation solver, Numerical programs, searching genome sequence, quantum computer simulation and many more (SPEC 2006)

World of Benchmarks

CloudSuite (https://www.cloudsuite.ch/)

CloudSuite is a benchmark suite for cloud services. The benchmarks are based on real-world software stacks and represent real-world setups.

PARSEC (https://parsec.cs.princeton.edu/)

Benchmark suite composed of multithreaded programs. The suite focuses on emerging workloads and was designed to be representative of next-generation shared-memory programs for chip-multiprocessors.

World of Benchmarks

MobileBench (https://mobilebench.engineering.asu.edu/) comprising a selection of representative smart phone applications.

Many more application domain specific: Graph processing, ML perf,

Rules for Measuring Performance with Benchmarks

- No Source code modifications are allowed, or it is impossible
- Use one compiler, one language for all the benchmark programs and don’t play with the flags

- Else, you can cheat!!!
- Also, use *benchmark suites*, not a single benchmark

Pitfalls of Benchmarks

Benchmark not representative of all

Your workload is I/O bound → SPECCPU is useless

Benchmark is too old

Need to be periodically refreshed

Non-benchmarks

- Application kernels: A small code fragment or part of the program
- Synthetic benchmark : Not part of any real program!!
- Micro-benchmark

World of Simulators

- Functional Simulator: Used to verify the correct execution of the program. Can not be used for performance evaluation.

- Performance simulators:

(i) Trace-driven: ChampSim (https://github.com/ChampSim/ChampSim)

(ii) Execution-driven: gem5, Multi2sim

Functional simulator is part of the performance simulators.

Evaluation Continued

Pick a *relevant* benchmark suite

Measure IPC of each program

Summarize the performance using:

- Arithmetic Mean (AM)
- Geometric Mean (GM)
- Harmonic Mean (HM)

Which one to choose?

Example

	IMTEL	ABM	AND
App. one	10	20	30
App. two	20	30	40
App. three	30	40	10

Which machine performs better over IMTEL and why?

Example

	ABM	AND
App. one	2	3
App. two	1.5	2
App. three	1.3	0.3
A.M.	1.60	1.76
G.M.	1.57	1.21
H.M.	1.54	0.72

AM on ratios

	X	Y
App. 1	1	100
App. 2	1000	10

Normalized to X	X	Y
App. 1	1	100
App. 2	1	0.01
AM	1	50.005

Y is 50 times faster than X

Normalized to Y	X	Y
App. 1	0.01	1
App. 2	100	1
AM	50.005	1

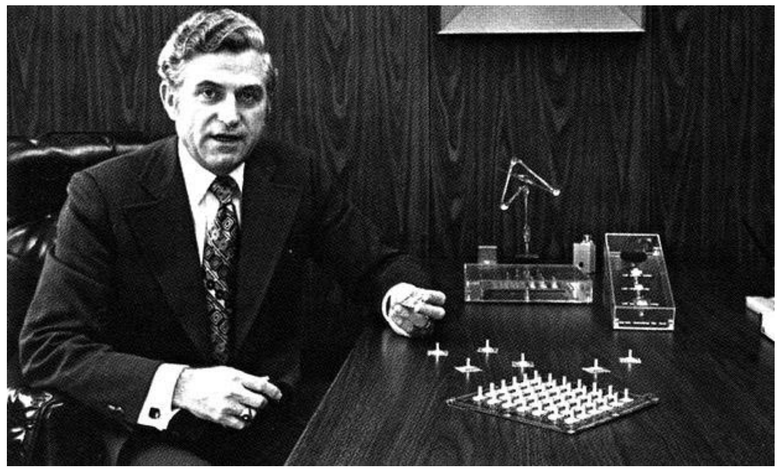
X is 50 times faster than Y

AM vs. GM

- GM of ratios is same as the ratio of the GMs
- Due to the aforementioned fact, the choice of reference does not matter if you go with GM.

Principles of Computer Design

Amdahl's Law (common case fast)

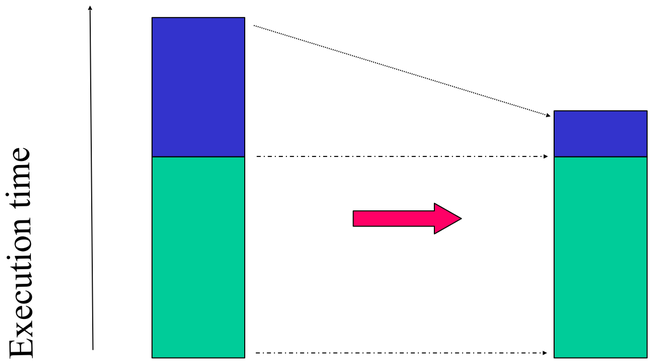


$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution Time}_{\text{old}}}{\text{Execution Time}_{\text{new}}}$$

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

Amdahl's Law and Speedup

- Speedup: How much faster a machine will run due to an enhancement?
- Need to consider two things while using Amdahl's law:
  - 1st… **Fraction of the computation time that can use the enhancement**
    - If a program executes in 30 seconds and 15 seconds of exec. uses enhancement, fraction = ½
  - 2nd… **Improvement gained by enhancement**
    - If enhanced task takes 3.5 seconds and original task took 7secs, we say the speedup is 2.



Amdahl's Law: Example

- Floating point instructions improved to run 2 times faster.
- But, only 10% of actual instructions are FP

  - $ExTime_{new} = ?$
  - $Speedup_{new} = ?$

Example: Answer

- Floating point instructions improved to run 2X faster.
  - But only 10% of actual instructions are FP.

$$\text{ExTime}_{new} = \text{ExTime}_{old} \times (0.9 + 0.1/2) = 0.95 \times \text{ExTime}_{old}$$

$$\text{Speedup}_{overall} = \frac{1}{0.95} = 1.053$$

Example 2

A common transformation required in graphics processors is square root. Implementations of floating-point (FP) square root vary significantly in performance, especially among processors designed for graphics. Suppose FP square root (FSQRT) is responsible for 20% of the execution time of a critical graphics benchmark. One proposal is to enhance the FSQRT hardware and speed up this operation by a factor of 10. The other alternative is just to try to make all FP instructions in the graphics processor run faster by a factor of 1.6; FP instructions are responsible for half of the execution time for the application. The design team believes that they can make all FP instructions run 1.6 times faster with the same effort as required for the fast square root. Compare these two design alternatives.

Example 2

We can compare these two alternatives by comparing the speedups:

$$\text{Speedup}_{\text{FSQRT}} = \frac{1}{(1-0.2)+\frac{0.2}{10}} = \frac{1}{0.82} = 1.22$$

$$\text{Speedup}_{\text{FP}} = \frac{1}{(1-0.5)+\frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23$$

Improving the performance of the FP operations overall is slightly better because of the higher frequency.

Amdahl's Law

Which one will provide better overall speedup?

A. Small speedup on the large fraction of execution time.
B. Large speedup on the small fraction of execution time.
C. Does not matter.

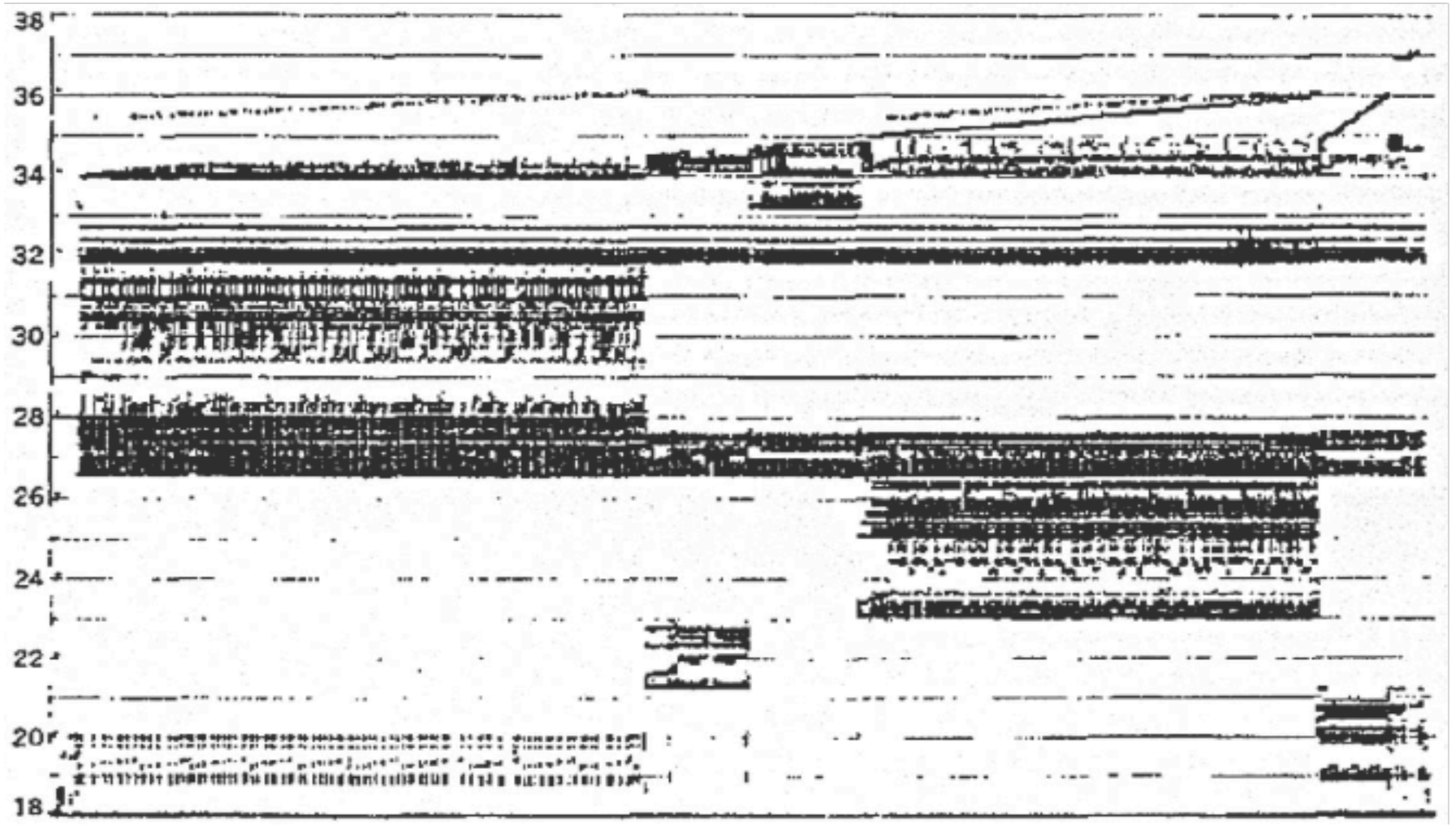
Depends on the difference between small and large. Mostly it is A.

Principle of Locality

- Programs tend to use the data and instructions they have used recently.

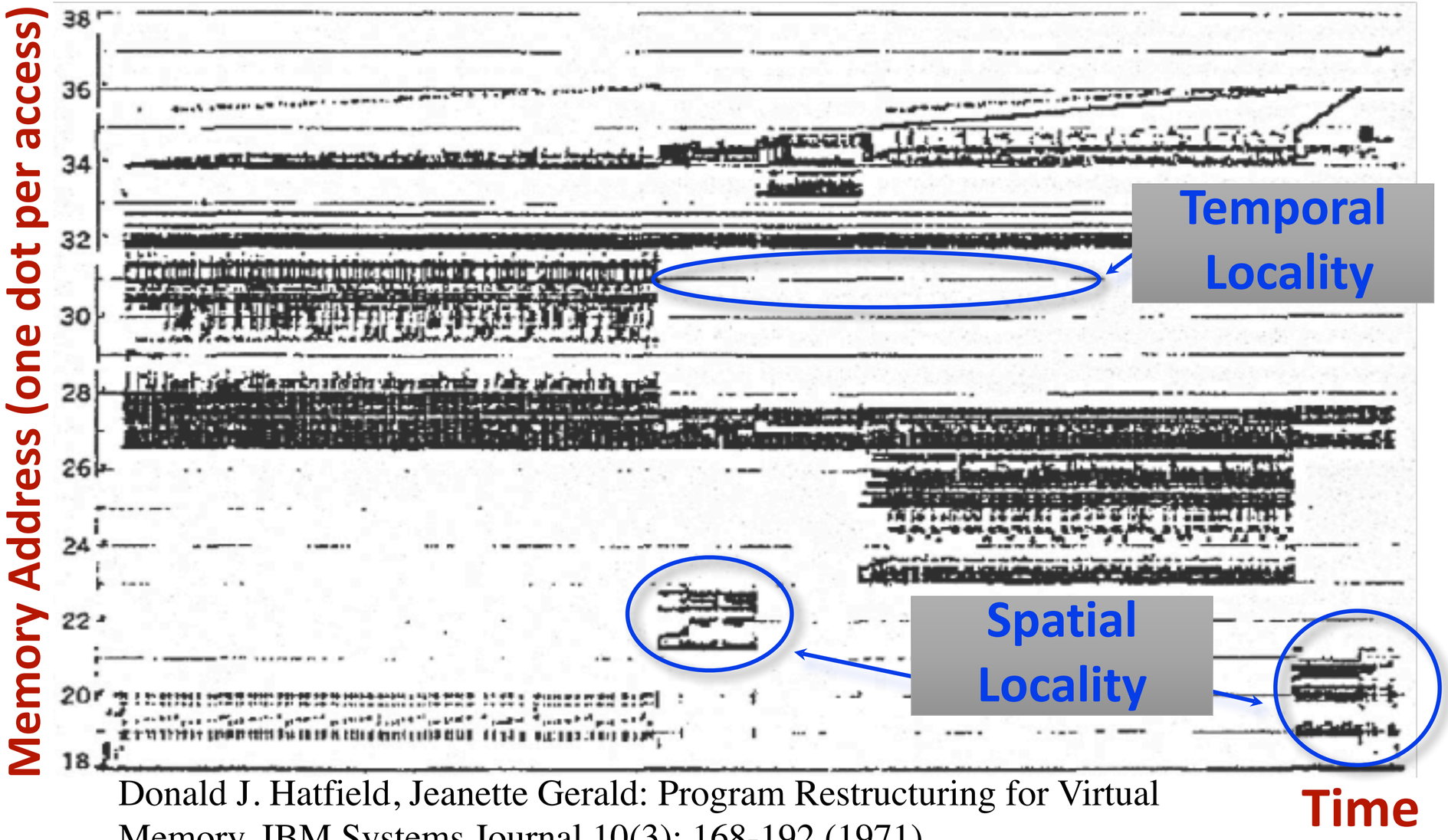
- So from the recent past, we can have a good idea of future
- Temporal Locality: Recently accessed items are to be used in near future.
- Spatial Locality: Items whose addresses are near to each other tend to referenced close together in time.

Let's look at the Applications (benchmarks)



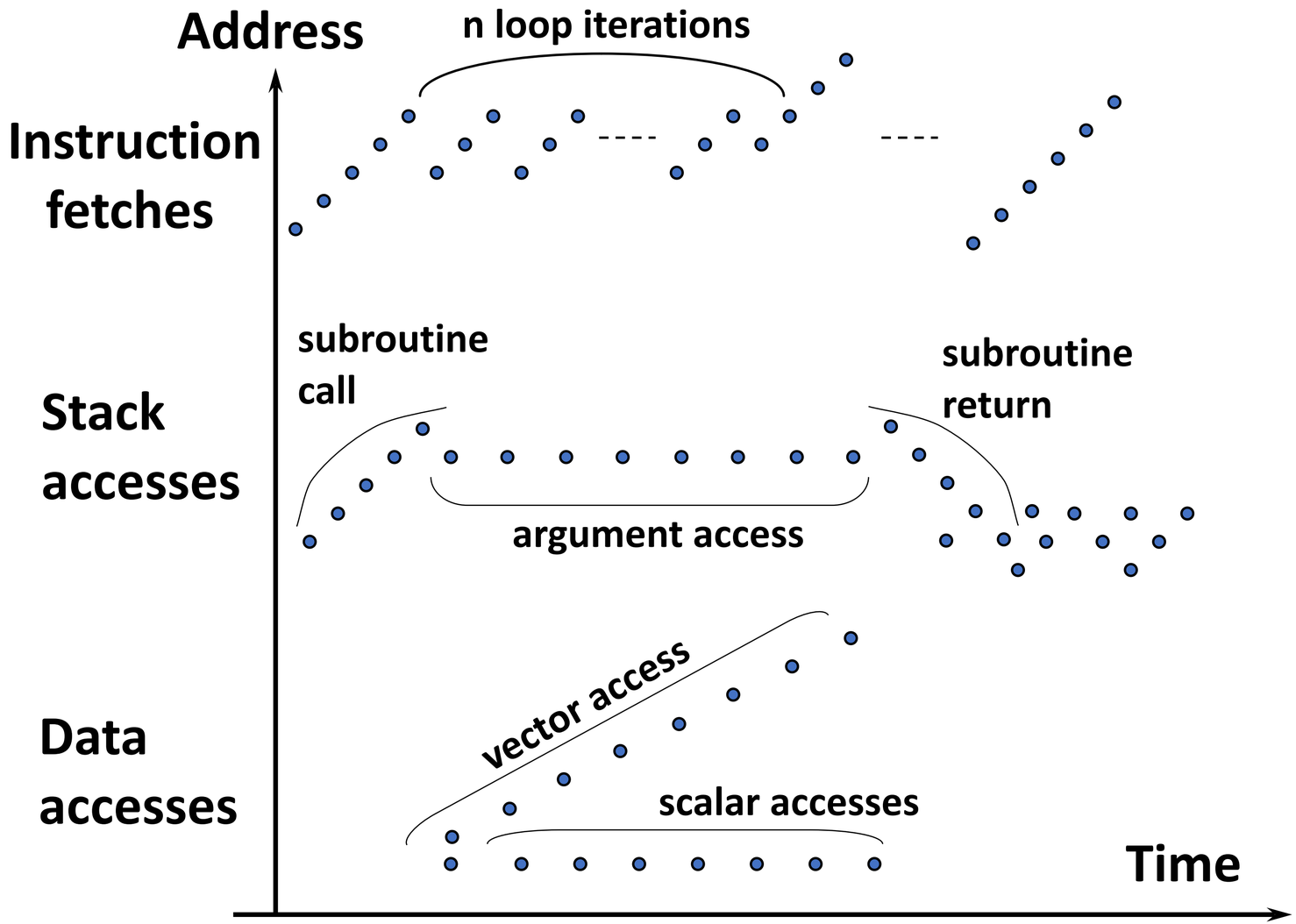
Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal 10(3): 168-192 (1971)

Locality



Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal 10(3): 168-192 (1971)

Few Examples



Back to Memory

World with no caches

North pole ☹

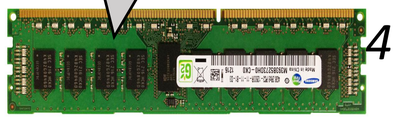
Core

32-bit Address

Data

200 to 300 cycles

Costly DRAM accesses☹



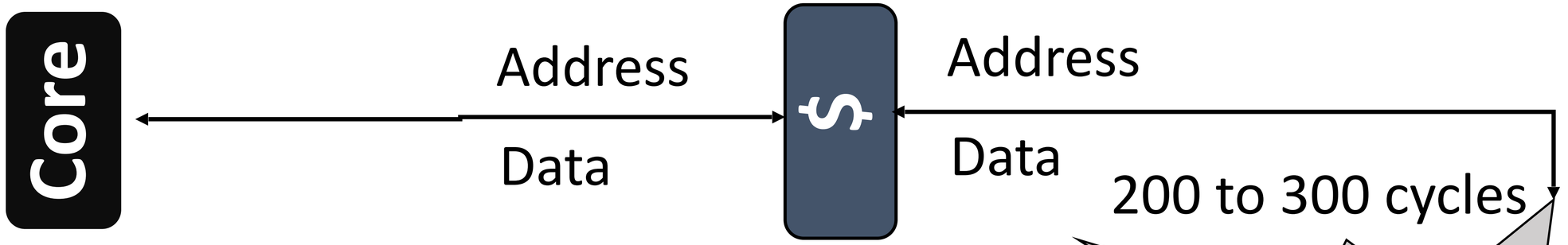
4 GB DRAM

South pole ☹

Minimizing costly DRAM accesses is critical for performance

Caching: Why does it works…?

North pole ☺



Costly DRAM accesses☹

Caching is a speculation technique ☺
Works – if locality

Do not ignore the common case

Reduction in DRAM accesses ~ Improvement in execution time

WRONG!

What if your program is not memory intensive

Book

P & H, Chapter 4