Computer Architecture, Lecture 2: Measure your performance

Hossam A. H. Fahmy

Cairo University

Electronics and Communications Engineering

Performance

Performance is usually measured using one of two ways:

latency: the time delay taken to finish the task or

throughput: the number of tasks finished in a fixed time.

Are they the inverse of each other? If not are they completely independent?

Real life examples:

- baking bread,
- passengers in a train station, or
- moving to and from the university using a bus versus a car.

Depending on the application for which we optimize the design we choose the appropriate measure. (Usually throughput.)

How do we measure time?

When we evaluate a processor we should be measuring how long it takes to complete some programs.

Real applications: Accurate but may not be portable to other processors for comparisons.

Some (modified) applications: Portable but may not be exactly what you run. Most benchmarks today fall in this category.

Kernels: those are key pieces from real programs. They are good to evaluate the performance of specific features but do not represent the complexities of real applications.

In the past people used some “toy” and “synthetic” benchmarks. These are even more removed from real applications and had serious flaws.

SPEC

- The Standard Performance Evaluation Corporation is a consortium of companies to maintain a standardized suite of benchmarks. (SPEC: http://www.spec.org)
- There are benchmark suites for CPU, Graphics, JAVA, Web, Mail, . . .
- The most current CPU benchmark is CPU2017 with 20 integer programs and 23 floating point programs.
  - Integer programs: 10 C, 8 C++, and 2 Fortran.
  - Floating Point: 6 Fortran, 6 C, 2 C++, 5 (C and Fortran), 2 (C and C++), and 2 (C, C++, and Fortran).

Transaction processing

- The Transaction processing Performance Council is a consortium of companies to maintain benchmarks for transaction processing and databases. (TPC: http://www.tpc.org)
- A typical “transaction” refers to a database update for things such as inventory control, airline reservations, or banking. It includes disk read/write, operating system calls, network transfers, . . .
- The TPC benchmarks include the total response time (not just the rate) and the cost of the system.

Compare and summarize

Which one is better?

	A	B	C
Time for program 1	1	10	20
Time for program 2	1000	100	20
Total time	1001	110	40
A as reference	1	10/0.1	20/0.02
B as reference	0.1/10	1	2/0.2
C as reference	0.05/50	0.5/5	1

A few important questions:

- How often is each program run?
- Are we considering one machine as the base and comparing the others to it?
- Are we interested in the latency or the throughput?

Different “means” mean different things!

Arithmetic mean: Each latency is multiplied by a weighting factor indicating the relative frequency of the program in the workload. Average is $\sum_{i=1}^{n} w_i t_i$. Good for quantities that have units such as time and that might be added.

Geometric mean: Good for quantities that are ratios without any units. The average is $\sqrt[n]{\prod_{i=1}^{n} \text{ratio}_i}$.

Harmonic mean: For throughput, the average rate is

$$\frac{n}{\left(\sum_{i=1}^{n} \frac{1}{\text{rate}_i}\right)}.$$

Moving at 10 km/h for 1 km and then 30 km/h for 1 km does not give an average of 20 km/h but rather $2/(\frac{1}{10} + \frac{1}{30}) = 15$ km/h.

Pros and Cons

	A	B	C		Geometric mean	
Time for program 1	1	10	20			
Time for program 2	1000	100	20			
Total time	1001	110	40			
A as reference	1	10/0.1	20/0.02	1	1	0.63
B as reference	0.1/10	1	2/0.2	1	1	0.63
C as reference	0.05/50	0.5/5	1	1.58	1.58	1

The geometric mean is consistent regardless of which machine is the reference. However, are A and B really equal?

Guiding principles

There are a few important principles that help architects produce better systems.

- Make the common case fast. If most of your code runs faster the whole program becomes faster.
- Exploit locality. This is the basic principle for caching.
- Exploit Parallelism. Parallel execution essentially improves the throughput.

What is the speedup? (Amdahl's law)

We define the speedup as the ratio between the original time taken and the time taken with an enhanced (usually parallel) execution.

$$S_p = \frac{T_1}{T_p}$$

If we can improve 75% of the program to take only 1/3 of the time then the speedup is

$$S_p = \frac{T_1}{0.25T_1 + \frac{0.75T_1}{3}} = 2$$

and not 3!
For 30% and 1/3 we get $S_p = \frac{T_1}{0.70T_1 + \frac{0.30T_1}{3}} = 1.25$.
For 75% and $1/\infty$ we get $S_p = \frac{T_1}{0.25T_1 + \frac{0.75T_1}{\infty}} = 4$.

How long does my program take?

The time a program takes is

$$\begin{aligned}\text{CPU time} &= \text{Dynamic instruction count} \\ &\quad \times \text{average CPI} \\ &\quad \times \text{Clock cycle time.}\end{aligned}$$

Always look at the three components not just one of them!
What happens if one instruction is waiting for the operand to arrive from the memory?

The upshot

- Make the common case fast but remember that the uncommon case eventually sets the limit.
- You must have a balanced system where the resources are distributed according to where time is spent.
- Your system's performance must be above the required average! The peak will be reduced by dependencies and memory stalls.

Shall we ignore other things besides speed?

Certainly NO.

- Real companies sell their products for profit. Customers pay to buy. The price is a very important metric. For CPUs, we often look for the best performance/price.
- For embedded systems that are battery powered, the performance/watt is an often quoted metric.
- The energy consumed is a major contributer to the running costs. The Total Cost of Ownership (TCO) is the sum of Capital Expenses (CapEx) and Operational Expenses (OpEx). DNN designs optimize performance/TCO and target a certain Service Level Objective (SLO).