CMP1203

Lecture 10

Performance

- is the amount of work accomplished by a computer system.
- Factors to take into account when determining performance;

Technology

- circuit speed (clock, MHz)
- processor technology (how many transistors on a chip)

Organization

- type of processor (ILP)
- configuration of the memory hierarchy
- type of I/O devices
- number of processors in the system

Software

- quality of the compilers
- organization & quality of OS, databases, etc.

Performance

Why study performance metrics?

- determine the benefit/lack of benefit of designs
- computer design is too complex to understand performance & performance bottlenecks
- have to be careful about what you mean to measure & how you measure it

We are interested in answering questions like:

Why is some hardware better than others for different programs?

What factors of system performance are hardware related? (e.g., Do we need a new machine, or a new OS?)

How does the machine's instruction set affect performance?

Factors affecting Performance

- **The clock rate**

The speed at which a processor is clocked plays an important role in its performance; that's why faster chips (of the same type and generation) are more expensive than slower chips. However, a 3 GHz chip from manufacturer P might run a program faster than a 3.5 GHz chip from manufacturer Q. You cannot compare clock rates between different chips. This is because different chips perform operations differently (for example, the ALU of one computer is different to the ALU of another computer).

Factors affecting Performance

- **The chip architect**

Different families of chips (e.g., ARM or Intel iCore or AMD) have different architectures or instruction sets. Consequently, they execute code in different ways using different numbers of registers, different instruction types and different addressing modes. Some architectures might favour arithmetic processing operations whereas other might favour graphical operations. You cannot easily compare chips with different architectures.

- **The chip microarchitecture**

The microarchitecture of a chip (also called its organization) is its structure (implementation) and defines the way in which the architecture (instruction set) is implemented in silicon at the level of buses, registers and functional units. Over the years, Intel's IA32 architecture has remained largely constant, whereas great advances have been made in its microarchitecture leading to much faster processors

Factors affecting Performance

- **Number of cores**

The performance of multicore processors is strongly dependent on the type of program being executed because some software can take advantage of parallel processing, whereas other software cannot be parallelized. Consequently, you may have a chip with four CPUs, but only one or two of them is actually in use for most of the time.

- **Cache memory**

Cache memory holds frequently used data and instructions and is far faster than external (i.e., off the chip) main DRAM memory. The quantity of cache memory and its structure has a great impact on system performance. Cache memory is necessary because, over the years, the performance of DRAM has increased more slowly than the performance of CPUs. Consequently, there is a bottleneck between processors and memory which cache memory helps to alleviate.

Factors affecting Performance

- **Main memory**

The main memory is composed of DRAM chips. These have gradually been improved over the years; for example, DRAM architectures have changed from DDR to DDR4 (in 2014).

- **Secondary storage**

Secondary storage, hard drives and now SSDs (solid state disks) hold data not currently in main memory. Hard drives are very slow compared to main store and can reduce the performance of a computer if frequent access to secondary memory is necessary. Newer solid state disks use semiconductor memory to store data and are far faster than hard disks.

- **General**

Many other factors determine performance; for example buses that distribute information between memory, the CPU and peripherals, chip sets that connect processors to buses and handle input/output transactions.

- **Software**

The performance of a computer system depends on it software as well as its hardware. Processors execute machine code. Compilers translate high-level code into machine code. Consequently, a good compiler may be able to generate much faster code than a poor compiler. It is dangerous comparing machines that have used different compilers because you might be comparing compiler efficiency rather than machine performance.

Metrics for computer performance

Raw speed: peak performance (never attained)
Execution time: time to execute one program from beginning to end
• the “performance bottom line”
• wall clock time, response time
• Unix time function: 13.7u 23.6s 18:27 3%
Throughput: total amount of work completed in a given time
• transactions (database) or packets (web servers) / second
• an indication of how well hardware resources are being used
• good metrics for chip designers or managers of computer systems
(Often improving execution time will improve throughput & vice versa.)

Metrics for computer performance

- Metrics for computer performance; clock rate, MIPS, Cycles per instruction, benchmarks

Clock Rate

- Instead of reporting execution time in seconds, we often use cycles
- clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec)
- It is measured in clock cycles per second or its equivalent, the SI unit hertz (Hz), the clock rate of the first generation of computers was measured in hertz or kilohertz (kHz the speed of modern CPUs is commonly advertised in gigahertz (GHz).

Metrics for computer performance

MIPS

- MIPS means ***millions of instructions per second***. Unlike clock rate, MIPS provides some idea of the work actually performed.
- However, even MIPS is not a good indication of performance because not all instructions perform the same amount of computation.
- The greatest weakness in this metric is that different machine architectures often require a different number of machine cycles to carry out a given task. The MIPS metric does not take into account the number of instructions necessary to complete a specific task.

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

Metrics for computer performance

- The most evident contrast can be seen when we compare RISC systems to CISC systems.

Type A

```
Loop   LDR  r0,[r1]
       ADD  r1,r1,#1
       STR  r0,[r2]
       ADD  r2,r2,#1
       SUB  r3,r3,#1
       BNE  Loop
```

Type B

```
Loop  MOV  [r1]+,[r2]+
      DECB r3,Loop
```

Metrics for computer performance

- Type A code is like typical **RISC code** (reduced instruction set code) that supports register-to-register operations and the only memory access instruction are load and store. In this code we use two pointer registers, r1 and r2, and update them on each pass round the loop. This fragment of code requires six instructions per iteration.
- Type B code is typical **CISC code** (complex instruction set) code that supports memory-to-memory operations. For example, the instruction MOV [r1]+,[r2]+ reads the contents of the memory location pointed at by register r1 and stores the data from memory in the memory location pointed at by register r2. Then, both pointer registers r1 and r2 are incremented. That's a lot of work performed by one instruction. The DECB is a decrement and branch instruction that implements a looping mechanism.
- Suppose a type A processor has a MIPS rating of 1,000, and a type B processor has a MIPS rating of 500. The type A processor appears twice as fast as the type B processor. However, in this example the type A processor will have to execute three times as many instructions as a type B processor. If a type A processor requires 1 unit of time to perform the operation, the type B processor operating at half the MIPS will require 2 units of time to perform the same number of instructions. Since processor B requires only 1/3 the number of instruction, the actual time taken by processor B will by 2/3 the time taken by processor A. In this case, processor B operating at only half the number of MIPS is faster than processor A. The MIPS rating does not help us to compare processors with different architectures.

Metrics for computer performance

MFLOPS

- MFLOPS is another measure of performance. It means millions of floating-point operations per second and is generally used to characterize the performance of computers dedicated to scientific and numerical applications.
- MFLOPS suffers from the same limitations as MIPS (i.e., it measures the number of operations and not the amount of work performed); however, MFLOPS does at least indicate the amount of useful calculations performed by the computer.
- MFLOPS is an unreliable metric when comparing different machines because not all computers implement floating-point arithmetic in the same way; for example, a particular computer may not have hardware floating-point division instructions which means that division must be carried out in software using a lot of primitive floating-point operations. Similarly, some floating-point systems generate trigonometric and similar functions in hardware and some in software. An MFLOPS rating only tells us about the total number of floating-point operations and not how much work they perform.

Metrics for computer performance

CPI

- CPI or clock cycles per instruction. This metric gives an indication of the average number of clock cycles it takes to execute each instruction.
- is used by those studying computer architecture to evaluate different implementations of the same architecture or to compare compilers.

CPU Clock Cycles = Number Of Instructions * CPI

Metrics for computer performance

Example; a computer's instruction set can be divided into four categories: arithmetic and logical operations (ALU), load from memory, store in memory, and flow control (Branch). The following table gives the number of clock cycles taken by each of these categories of instruction, and the relative frequency of the instruction class in a program.

Class	Frequency	Cycles per instruction
ALU	55%	1
Load	20%	10
Store	10%	4
Branch	15%	3

Metrics for computer performance

- this data can be used to calculate the total number of cycles taken by each instruction class by multiplying the relative frequency by the number of cycles per instruction. For example, loads take 10 cycles and are executed in 20% of the program. The relative number of cycles taken by the load is therefore 10 x 20% = 2 cycles. If we add the relative number of cycles for each instruction (in this case 3.40) we get the average number of cycles per instruction for the program being executed.

Class	Frequency	Cycles per instruction	Total cycles	Time
ALU	55%	1	1*0.55=0.55	0.50/3.40=14.7%
Load	20%	10	10*0.20=2.00	2.00/3.40=58.8%
Store	10%	4	4*0.10=0.40	0.40/3.40=11.8%
Branch	15%	3	3*0.15=0.45	0.45/3.40=13.2%
Total	100%		3.40 cycles/instruction	

Evaluation of Performance Metrics

- The usual way of comparing processors is by means of benchmarks, or test programs designed to give an indication of the performance of a computer.
- Benchmarks can be divided into three categories:

1. Kernels
2. Toy benchmarks
3. Synthetic benchmarks

Benchmarks

- Kernels are short segments of real code (i.e., actual code) that are used to evaluate performance. Essentially, about 10 percent of code performs about 80% of the computation and kernels rely on testing that 10% of the code that is frequently used. In general, kernels are not good indicates of overall performance but they do help researches to investigate key aspects of a computer system such as cache performance.

- Toy benchmarks are short programs that can be used to measure performance; for example, the quicksort algorithm. Such benchmarks are of limited use in today's world with fast high-performance computers.

- Synthetic benchmarks are programs designed to test computers by providing the type of code that would be run in practice; that is, a synthetic benchmark is an artificial program designed to match the intended profile of the computer. Typical examples of synthetic benchmarks are Dhrystone and Whetstone (although these are little used today).

Benchmarks

- Early benchmarks included Whetstone (floating-point and numerical computation), Dhrystone (uses a lot of integer string operations), and Linpack. These are also called synthetic benchmarks because they run artificial code designed for benchmarking rather than real world applications.
- Old benchmarks like Whetstone, Dhrystone and Linpack are not very popular today and suffer from a number of limitations; for example:

1. They are small (in terms of the size of code) and can fit in cache memory which gives them an unfair advantage because a large program will not generally fit in cache memory.
2. The type of code tested is relatively old-fashioned and does not describe well modern computer operations
3. They are prone to compiler tricks; that is, compilers can be designed to optimize benchmarks.
4. Because they are such small programs, their run time on modern machines is very short which means that the accuracy of the benchmark is limited.

Benchmarks

SPEC

- The Standard Performance Evaluation Corporation (SPEC) that was founded in 1988 by a consortium of computer companies including Apollo, Hewlett-Packard, MIPS Computer Systems, and SUN Microsystems. The role of SPEC is to provide the industry with a realistic yardstick to measure the performance of advanced computer systems. SPEC has created a standardized set of application-oriented programs to be used as benchmarks.
- Every few years SPEC updates the suite of programs it uses in its benchmarks to take account of developments in computing. SPEC also provides specific benchmarks to test network-based computers and high-performance graphics based systems. SPEC benchmarks are respected because they are used by industry and academia and it is thought that they do not favour a single architecture because the SPEC consortium includes members from competing companies and they all keep watch on each other.

Use of benchmarks

- Benchmarks enable users to buy computers with the appropriate cost: performance ratio.
- Benchmarks show how computing is progressing by showing the increase in performance as a function of time
- Benchmarks help manufactures create faster machines
- Benchmarks help the research community by providing a means of measuring the performance change when systems are modified.

Averaging Metrics

Arithmetic: (1/N) * ∑P=1..N Latency(P)

• For units that are proportional to time (e.g., latency)

Harmonic: N / ∑P=1..N 1/Throughput(P)

• For units that are inversely proportional to time (e.g., throughput)

• You can add latencies, but not throughputs

Latency(P1+P2,A) = Latency(P1,A) + Latency(P2,A)

Throughput(P1+P2,A) != Throughput(P1,A) + Throughput(P2,A)

1 mile @ 30 miles/hour + 1 mile @ 90 miles/hour

Average is not 60 miles/hour

Geometric: N√∏P=1..N Speedup(P)

• For unitless quantities (e.g., speedup ratios)

Averaging Metrics

- Latency (execution time): time to finish a fixed task
- Throughput (bandwidth): number of tasks in fixed time

Other forms of Performance

- **Reliability**

If you store large amounts of data on a computer you do not wish to lose it in a system crash. Similarly, if you use your computer to move the control rods into and out of a nuclear pile you don't want to encounter a fault that suddenly withdraws all control rods at the same time.

- **Power**

Probably the most important metric today (after performance) is power consumption. Computers use energy and that is a valuable resource. It is expensive. Moreover, once you have used it, electrical energy ends up as waste heat and you have to use even more energy to get rid of the heat – cooling and air conditioning. Heat also reduces the lifetime of components and computers.

Today, the power consumption of electrical devices is a prime parameter and already manufacturers are beginning to stress the energy efficiency of their products.

- **Latency**

Latency can be defined as the time between triggering an event and the time at which a result is achieved. In other words, latency is a start-up delay or a transport delay. A good example, of latency in computing is the start-up delay when the computer is first booted. Latency should not be confused with performance; even a fast computer may have a long start-up latency.

Amdahl’s Law

- This law states that the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used.
- the overall speedup of a computer system depends on both the speedup in a particular component and how much that component is used by the system.

S is the overall system speedup;

f is the fraction of work performed by the faster

k is the speedup of a new component.

$$S = \frac{1}{(1 - f) + f/k}$$

Amdahl’s Law

- **Example;**

If most of your daytime processes spend 70% of their time running in the CPU and 30% waiting for service from the disk. Suppose also that someone is trying to sell you a processor array upgrade that is 50% faster than what you have and costs $10,000. The day before, someone called offering you a set of disk drives for $7,000. These new disks promise to be 150% faster than your existing disks. You know that the system performance is starting to degrade, so you need to do something. Which would you choose to yield the best performance improvement for the least amount of money?

Amdahl's Law

$$f = .70,\ k = 1.5,\ \text{so } S = \frac{1}{(1 - 0.7) + 0.7/5} = 1.30$$

We therefore appreciate a total speedup of 1.3 times, or 30%, with the new processor for $10,000.

For the disk option, we have:

$$f = .30,\ k = 2.5,\ \text{so } S = \frac{1}{(1 - 0.3) + 0.3/2.5} \approx 1.22$$

For the processor option, we have:
We therefore appreciate a total speedup of 1.3 times, or 30%, with the new processor for $10,000.

For the disk option, we have:
The disk upgrade gives us a speedup of 1.22 times, or 22%, for $7,000.

Amdahl’s Law

- All things being equal, it is a close decision. Each 1% of performance improvement resulting from the processor upgrade costs about $333. Each 1% with the disk upgrade costs about $318. This makes the disk upgrade a slightly better choice, based solely on dollars spent per performance improvement percentage point. Certainly, other factors would influence your decision. For example, if your disks are nearing the end of their expected lives, or if you’re running out of disk space, you might consider the disk upgrade even if it were to cost more than the processor upgrade.

- In summary, it is pointless to apply resources to parts of a system whose contribution to overall performance is small

References

Linda Null and Julia Lubor. The Essentials of Computer Organization and Architecture, Second Edition, Jones & Bartlett. ISBN-10: 0763737690, ISBN-13: 978-0763737696 (Chapter 11)