HPC :Tutorial 01

PERFORMANCE EQUATION I

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

equivalently

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

How to improve performance ?

- reduce the number of cycles for a program
- reduce the clock cycle time, or, equivalently,
- increase the clock rate
-

https://bheemhh.github.io/HPC24.html

Example

- Our favorite program runs in 10 seconds on computer A, which has a 400MHz. clock. Calculate the number of clock cycles.
- We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds.
- The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program.

- What clock rate should we tell the designer to target?

Solutions

Assume the program takes X cycles and the desired rate is Y

(X/ 400/10^6) /(1.2X/Y) = 10/6

therefore Y = 800 MHz

https://bheemhh.github.io/HPC24.html

- *cycle time* (seconds per cycle)
- *clock rate* (cycles per second)
- **(*average*) *CPI* (cycles per instruction)**
  - a floating point intensive application might have a higher average CPI

https://bheemhh.github.io/HPC24.html

PERFORMANCE EQUATION II

CPU execution time for a program = Instruction count X average CPI X Clock cycle time

- *Derive the above equation from Performance Equation I*

- CPU execution time = IC * CPI * Clock cycle time
  - IC : Instruction count
  - *CPI : Clock cycles per instruction*

https://bheemhh.github.io/HPC24.html

CPI EXAMPLE I

- Suppose we have two computers
  - Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program
  - Computer B has a clock cycle time of 500 ps and a CPI of
  - 1.2 for the same program.
- Which machine is faster for this program, and by how much?

We know that each computer executes the same number of instructions for the program; let's call this number I. First, find the number of processor clock cycles for each computer:

$$\text{CPU clock cycles}_A = I \times 2.0$$
$$\text{CPU clock cycles}_B = I \times 1.2$$

Now we can compute the CPU time for each computer:

$$\begin{aligned}\text{CPU time}_A &= \text{CPU clock cycles}_A \times \text{Clock cycle time} \\ &= I \times 2.0 \times 250 \text{ ps} = 500 \times I \text{ ps}\end{aligned}$$

Likewise, for B:

$$\text{CPU time}_B = I \times 1.2 \times 500 \text{ ps} = 600 \times I \text{ ps}$$

Clearly, computer A is faster. The amount faster is given by the ratio of the execution times:

$$\frac{\text{CPU performance}_A}{\text{CPU performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{600 \times I \text{ ps}}{500 \times I \text{ ps}} = 1.2$$

We can conclude that computer A is 1.2 times as fast as computer B for this program.

AMDAHL'S LAW

- the performance improvement to be gained from using some faster mode of execution is limited by the **fraction of the time** the faster mode can be used

$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

Alternatively,

$$\text{Speedup} = \frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$$

https://bheemhh.github.io/HPC24.html

AMDAHL'S LAW

- Performance Improvement depends on two factors:
  - The fraction of the computation time in the original computer that can be converted to take advantage of the enhancement
  - The improvement gained by the enhanced execution mode, that is, how much faster the task would run if the enhanced mode were used for the entire program

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

https://bheemhh.github.io/HPC24.html

AMDAHL'S LAW

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

The overall speedup is the ratio of the execution times:

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

https://bheemhh.github.io/HPC24.html

Example

1. Let a program have 40 percent of its code enhanced to run 2.3 times faster. What is the overall system speedup S?

Let a program have 40 percent of its code enhanced to run 2.3 times faster. What is the overall system speedup S?

$\text{Fraction}_{\text{Enhanced}} = 0.4$

$\text{Speedup}_{\text{Enhanced}} = 2.3$

$$\text{Speedup}_{\text{Overall}} = \frac{1}{(1-0.4)+\frac{0.4}{2.3}} = 1.292$$

Example

- Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time.
- *How much do we have to improve the speed of multiplication if we want the program to run 5 times faster?*

Example

- Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time.
- *How much do we have to improve the speed of multiplication if we want the program to run 5 times faster?*

$$\text{Execution time after improvement} = \frac{80 \text{ seconds}}{n} + (100 - 80 \text{ seconds})$$

Since we want the performance to be five times faster, the new execution time should be 20 seconds, giving

$$20 \text{ seconds} = \frac{80 \text{ seconds}}{n} + 20 \text{ seconds}$$

$$0 = \frac{80 \text{ seconds}}{n}$$

- Suppose we enhance a machine making all floating-point instructions run five times faster. The execution time of some benchmark before the floating-point enhancement is 10 seconds.
  - *What will the speedup be if half of the 10 seconds is spent executing floating-point instructions?*
- We are looking for a benchmark to show off the new floating-point unit described above, and want the overall benchmark to show a speedup of 3. One benchmark we are considering runs for 100 seconds with the old floating-point hardware.
  - *How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?*