Performance optimization

June, 2014

Presenter:
Adrian Loteanu
adrian.loteanu@intel.com

Disclaimer

- The views expressed in this tutorial are those of the people delivering the tutorial.
- We are not speaking for our employers.
- We are not speaking for any third party.

Legal Notice

- Intel and the Intel logo, are trademarks of Intel Corporation in the U.S. and/or other countries.

- Other names and brands may be claimed as the property of others.

- THE MATERIALS ARE PROVIDES “AS IS”, WITHOUT WARRANTY O ANY KIND, EXPRESS OR IMPLIED, INCLUDED BUT NOT LIMITED TO THE WARRANTIES OF MECHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN ACTION OF CONTRACT, TORT, OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE MATERIALS OR THE USE OR OTHER DEALINGS IN THE MATERIALS

- Creative Commons License: Attribution-NonCommercial-ShareAlike 4.0 International
  http://creativecommons.org/licenses/by-nc-sa/4.0/

Disclaimer

Some numbers have been made up for educational purposes.

Information in this presentation, especially regarding specific numbers, is merely made up for educational/learning purposes and should not be used for any other purpose. There is no guarantee that any information contained in the presentation is correct or accurate.

What exactly is performance? Which is "better"?

Means of transportation	Top Speed (km/h)	Fuel consumption (l/100km)	Capacity (people)	Throughput (cap x top speed)
Motorcycle	271	8	2	542
Car	429	15	5	2145
Bus	160	25	60	9600

*Numbers are fictional.

What exactly is performance? Which is "better"?

Processor category	Core Freq(MHz)	TDP (Watts)	Threads	Throughput (Threads x Freq)
Motorcycle	1200	8	2	2400
Car	3500	50	8	28000
Bus	2900	120	16	46400

*Numbers are ficitional.

What is performance?

- Performance is hard to define.

One way of looking at performance

$$Performance = \frac{1}{Execution\ \_time}$$

Example of performance measurement

Sample output from the Linux command “time” which measures application runtime in this case for a matrix multiplication application:

```
time java
MatrixMulDouble
real    0m3.321s
user    0m58.160s
sys     0m0.064s
```

- Notice that user time is larger than real time – this is because of multiple threads (think man-hours). CPU time is the user time divided by the # of threads.
- The most relevant measure of performance is ultimately **elapsed time** or **real time**.

*Data generated by running a toy matrix multiplication program on an unspecified CPU. Your results may differ.

Latency vs. Throughput

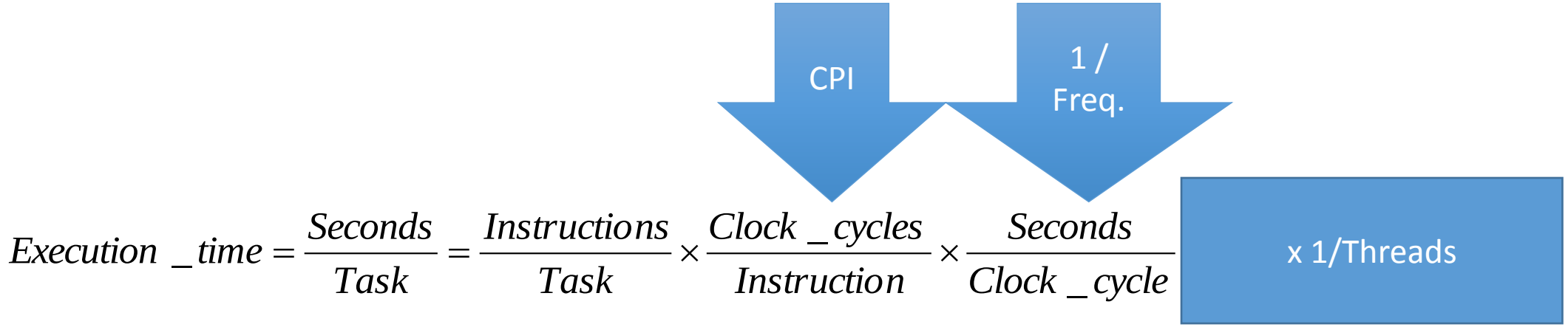
- **Important notions – throughput vs. latency:**
- **Latency – duration of a single task from start to end**
- **Throughput – number of tasks completed / time**

- **Performance can be defined in terms of latency of throughput or any complex combination of the 2 (max throughput with X% of response times < Y s).**

Performance definition

- Generally performance is defined in terms of execution time.
- Performance = 1 / Execution time (as measured by the users clock)
- Performance = Useful operations / execution time

The processor performance equation – single thread



A little about power

Dynamic Energy =~ ½ x Capacitive load x Voltage^2

Dynamic Power =~ ½ x Capacitive load x Voltage^2 x Frequency switched

Static Power (aka leakage) =~ Current(static) x Voltage

Performance optimization tricks – bit hacks

- Bit hacks are generally designed to use arithmetic and logic tricks to eliminate costly operations in terms of performance.

- Popular at programming interviews

Determine if an integer is a power of 2

- X = x && !(x & (x - 1));

Location of least significant 1:

- l = x & (-x);

Swapping 2 variables without an auxiliary

- Normally:
- Swap (a,b)
- temp = a
- a=b
- b=temp

Swapping 2 variables without an auxiliary

- a = a^b
- b = a^b
- a = a^b
- Note: ^ means xor

- If the algorithm does a lot of swapping it may remove the need to use an extra variable(register) and increase your throughput. Modern compilers can optimize really well so this optimization in particular does not have significant benefits if any.

Minimum/Maximum of 2 integers without using an “if”:

- “IF” statements or branches can be costly. Modern CPUs are pipelined. A branch would normally be evaluated near the end of the pipeline causing a stall – we don’t know which instructions to fetch next, do we get the “if” stream or the “else” one?
- Some modern CPUs try to predict branches and go on one path based on a prediction history. Very advanced branch predictors exist but some branches are just not predictable – like for example in a sorting algorithm where you have no way of knowing which numbers are bigger or smaller in advance.
- A miss-predicted branch causes a pipeline flush which can take dozens of clock cycles to recover or more. Could we sort without using branches?

Minimum/Maximum of 2 integers without using an “if”:

- Normally:

- If (x>y) { max = x; min = y; }
- Else {max=y; min = x; }

Minimum/Maximum of 2 integers without using an “if”:

- min = y ^ ((x ^ y) & -(x < y));
- max = x ^ ((x ^ y) & -(x < y));

- OR

- min = (x + y + abs(x-y)) / 2;
- max = (x + y – abs(x-y)) / 2;

- This is used especially in graphics where branches are particularly costly.