# Measuring Efficiency of Computation in Diverse Systems and Workloads

Isaac Sherman

# Abstract

We analyze 3 different systems under different tasks and gather the metrics necessary to calculate FLOPS/Watt in operations designed to emphasize 1) high arithmetic intensity processing and 2) memory-bound processing. Systems are a laptop with a 2.39 GHz Intel Core i7-4500U CPU, a desktop with a 3.30 GHz Intel Core i3-3220 CPU, and a Raspberry Pi 3b. We compare efficiency across tasks that emphasize variations in arithmetic intensity. Methodology is to generate no less than 10 minutes of performance data and then compare them across tasks and hardware. Laptop found to be most efficient use of energy, particularly when memory was not a limiting factor, but difficulties with data gathering in the Pi impair drawing other definitive conclusions.

# Introduction

The initial goal was to provide an analysis of FLOPS/Watt of 3 different systems, a laptop with a 2.39 GHz Intel Core i7-4500U CPU, a desktop with a 3.30 GHz Intel Core i3-3220 CPU, and a Raspberry Pi 3b. Unfortunately, the power meter recommended by the reference paper has apparently been lost in the mail, and we no longer expect it to arrive before the due date. Instead, we will be comparing the laptop and desktop using temperature and processor load monitors on periods when they are either compute or memory bound. We are forced to rely on the reports from onboard the chip because we failed to obtain the power meter. The Pi is the most problematic in this regard, as it can only report its temperature and possibly a partial voltage.

# Methodology

A note on FLOPS: FLOPS is Floating Point OPerations per Second. Flops is FLoating point OPerationS.

To find FLOPS per watt, we need to solve the equation

On the laptop and desktop, we have an internal measure of power. They are the onboard power sensors provided by the manufacturer, and may not be as reliable as using an external power meter, which was unavailable due to a shipping issue. On the Raspberry Pi, as we don’t have a direct measure of power, we must infer it using

Where T is the temperature (which we can measure with the vcgencmd command) and C is some conversion factor which we’ll have to assume. If power is I \* V, then we have an upper bound on the Pi based on its power supply, which is 2.4A \* 5.25V, or 12.6 Watts. We poll power on the laptop and desktop every 5 seconds. We do have voltage, but not amperage, so that gives us one piece of the equation- we may assume amperage remains constant and calculate power by multiplying by the voltage we get by 2.4 to approximate Power, or we may try to infer C from equation 2. We can approximate C from the data we gather with the direct measure. Given the data from the laptop and the desktop, it seems that solving equation 2 for C gives us some value between .2 and .3. Exactly which values we use to calculate this make a difference, particularly because temperature is a lagging indicator of power. Thus, the measurements we use are taken from after both power and temperature have been stable for a short amount of time. On the laptop, we get a value of .2, where temperature is 75° and the wattage is 15. On the desktop, the power is less stable, but if we take the average of the ratios whenever the temperature is ≥ 100°, we get a C of ≅ .28. This value holds for both memory bound and high arithmetic intensity tasks. On the laptop, the C is slightly higher on the memory bound task than the arithmetic task (.21 and .2, respectively). With a sample size of only 2 machines, it’s hard to say how to predict the behavior of the Pi. What could influence C? Several things, but one seems particularly likely here: ventilation. Superior ventilation should reduce C, by displacing the aggregated heat faster than it can be generated. Therefore, placing the machines in order of ventilation, we would expect the Pi, with the best ventilation, to have the lowest C, and the desktop the highest. Making this assumption, we use a C which does not break the upper limit based on the power source. .175 and .15 both exceed the 12.6W limit, and so somewhat arbitrarily we use .125. This gives us a slightly generous estimate for the power use of the Pi.

In both cases below, the programs run for a minimum of 10 minutes. We don’t interrupt them at 10 minutes exactly, however, because we wouldn’t be able to accurately calculate how many Flops had been completed- instead they finish all threads that are running when ten minutes pass. Our programs report how many iterations have been completed and how long they took. Then, FLOPS can be found:

(3)

Where F is the number of Flops in an iteration, I is the number of iterations and Program\_Num is the number of programs. The numerator gives us the total Flops, the denominator is the length in seconds of the longest running program. See the data sheets for more detail on calculations. We calculate power by taking the average of all reported values when the load of the processors is greater than 50%, which has historically been a good indicator that the programs were running (typical load is around 15% for the laptop, 0% for the desktop). Then we use equation 1.

### Memory Bound Tasks

For the memory-bound benchmark, we run threads which use 4 400x400 matrices. These matrices are populated by random reals ϵ [0, 1.0). They are then naïvely matrix multiplied, such that each matrix multiply requires 400 multiplications per row and then a summation of 400 products for each of 400 results. Therefore, each matrix multiply requires 400^3 = 64 million Flops, and each loop iteration launches 4 threads and thus requires 256 million Flops. In addition, we run 4 of these programs simultaneously to achieve a 100% load.

Initially, we had attempted to use an optimized library (lib-atlas via numpy) to calculate this benchmark- however, the resulting FLOPS could only be described as ludicrously excessive. The optimized libraries were “cheating”, and not calculating the matrices the naïve way. Therefore, more than 6000 iterations were being completed in a single 10-minute run with a matrix 5 times larger. In contrast, only 4 were completed in the naïve sense, which means the optimizations have improved execution speed by more than 3 orders of magnitude- and that’s impressive.

### High Arithmetic Intensity Tasks

For the high arithmetic intensity, we have threads which do two multiplies and one addition on 8 combinations of different random numbers. These get passed on to the next iteration for 10 million “steps”. This, thus, limits memory access because the only load necessary (in theory) is the initial randomization of the numbers, and they may then be carried in registers through successive steps. In addition, there are 8 threads per iteration of the loop. Each thread requires 10,000,000 \* 8 \* 3 = 240 million Flops. This only gives us about 35-40% processor utilization, so we also employ 4 instances of the python program itself. This pushes the processor load to 90-100% for the duration of the experiment. On the pi, we reduced the number of steps to 100,000 because it wasn’t finishing in a reasonable amount of time.

# Results

We ran both programs several times, and include a report of a typical run on each platform and program combination.

|  |  |  |  |
| --- | --- | --- | --- |
| Platform | Test | FLOPS/Watt | Comment |
| Desktop | Arithmetic | 588150.7 |  |
| Desktop | Memory | 253953.849 |  |
| Laptop | Arithmetic | 829728.02 | Most energy efficient |
| Laptop | Memory | 356581.37 |  |
| Raspberry Pi | Arithmetic | 180054.2 | Using estimate of C |
| Raspberry Pi | Memory | 177086.1 | Using estimate of C |
| Raspberry Pi | Arithmetic | 592855.1 | Using Amps\*Volts |
| Raspberry Pi | Memory | 580693.9 | Using Amps\*Volts |
| Raspberry Pi | Arithmetic | 276218.8 | Using Average of above |
| Raspberry Pi | Memory | 271405.4 | Using Average of above |

We’ve included several different estimates of the Pi’s power use. Interestingly, they are all close to each other, with the Arithmetic performing slightly better in all 3 modes. This seems to imply that the tasks are similar in difficulty in a way which they were not on the Laptop and Desktop, which may mean that some further modification should have been done to the programs before running them on the Pi. Further, the Pi’s onboard voltage is somewhat suspect, as it never rose above 1.5 volts. This gives it a rather rosy approximation. Overall, we feel a significant level of skepticism toward the Pi-related results is in order.

Otherwise, the data seems to support the intuition we had going in- the desktop is more powerful, but the laptop is more energy efficient. It was interesting to monitor their temperatures during peak use, and to see how hot the desktop could get- hot enough to seriously burn anyone touching it. Also, the relative stability of behavior across machines was somewhat remarkable.

Code, data, and this document may be found at <https://github.com/IsaacSherman/CS530ProjectSherman> for a limited time.

### Charts