Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 9, Number 1, pages 35–39, January 2020

Measurement of Performance and Energy Consumption of OpenCV Programs on Raspberry Pi

Kazumasa Kadota[1], Ittetsu Taniguchi[2], Hiroyuki Tomiyama[1]
[1] Graduate School of Science and Engineering, Ritsumeikan University, Japan
[2] Graduate School of Information Science and Technology, Osaka University, Japan

***Abstract*—Raspberry Pi is widely used tiny computer not only for educational purposes but also for IoT devices. Since IoT devices are required to operate with low power/energy consumption, efficient energy management is necessary. In this paper, we measure the performance and energy consumption of Open CV programs on Raspberry Pi. The OpenCV programs are executed under the different execution methods and various execution frequencies, and the characteristics of performance and energy consumption are analyzed.**

Keywords—Raspberry Pi, OpenCV, energy consumption

I. Introduction

Nowadays, computer vision is popular application on IoT devices. Computer vision often requires high computational power, but low energy consumption is also important for IoT devices because of its limited power supply due to the battery capacity.

Raspberry Pi is well known and widely used tiny computers. Raspberry Pi is used not only for educational purpose but also for recent IoT devices. In order to realize long operational time for the dedicated application such as computer vision, efficient power/energy management is important for Raspberry Pi based IoT devices. Then the measurement of performance and energy consumption is an important first step to realize such energy management technology.

Various researches have been reported regarding the measurement and analysis of power/energy consumption of Raspberry Pi [1, 2, 3, 4]. Reference [1] measured the power consumption of Raspberry Pi, and the results are compared with PC and smart phones. Reference [2] focused on the sensor network such that Raspberry Pi is used as sensor node. The power consumption of Raspberry Pi based sensor node was measured and analyzed. Reference [3] also focused on wireless sensor network such that Raspberry Pi is used as the gateway. The power consumption of Raspberry Pi based gateway was measured and analyzed under the various execution frequencies. Reference [4] constructed the power consumption model based on the regression analysis with real measured power consumption data.

In this paper, we measure the performance and energy consumption of computer vision applications on Raspberry Pi. We employ the following three benchmark applications which use OpenCV: circle detection, Kmeans, and face detection. OpenCV supports parallel processing on multi-core CPUs and it can be customized when building OpenCV. In this paper, we measure the performance and energy consumption of three types of configurations: parallel execution using OpenMP, parallel execution using pthreads, and single-thread execution. In addition, CPU execution frequency of Raspberry Pi is also changed from 400MHz to 1.2GHz to measure.

The rest of the paper is organized as follows: Section 2 explains the experimental environment. Section 3 and 4 show the measurement results of execution time and energy consumption of single and multiple programs, respectively. The discussion is given in Section 5 and this paper is summarized in Section 6.

II. Experimental Environment

A. Equipment and Software

In this paper, Raspberry Pi 3 Model B was used to measure the performance and energy consumption. Raspberry Pi 3 Model B has ARM Cortex-A53 (4 cores) and 1GB SDRAM operating at up to 1.2GHz. We used KKmoon voltage / current tester UM24C to measure energy consumption.

OpenCV 3.4 was built as following configurations using GCC 7.4.

- OpenMP
- pthread
- Single-thread

The following OpenCV sample programs were executed on Raspberry Pi with Ubuntu 18.04 OS.

- Circle detection by Hough transform
- Clustering by Kmeans method
- Face detection by classifier cascade using Haar-like features

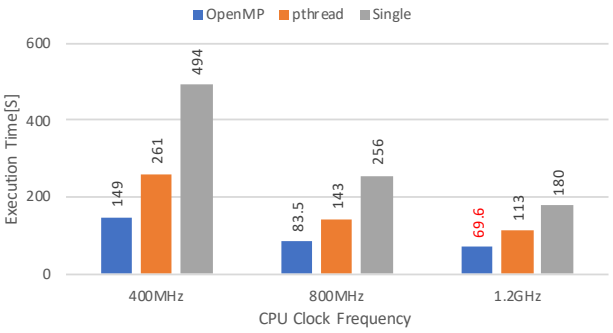
CPU clock frequency was fixed by setting Governor to Performance. We measured the performance and energy consumption under the following frequencies: 400MHz, 800MHz, and 1.2GHz.

B. Profiling

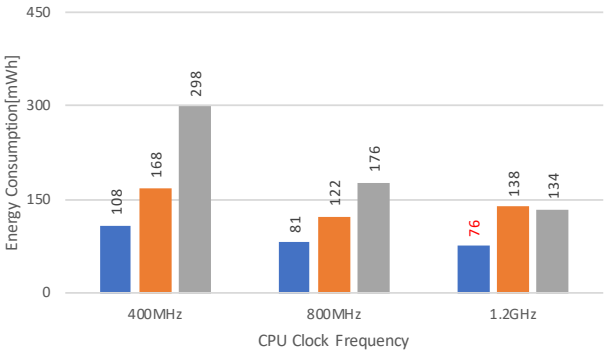
As a preliminary experiment, three programs were profiled. Three programs were executed on the Raspberry Pi using the OpenCV library built with three configurations: OpenMP, pthread, Single-thread. CPU clock frequency was fixed at 1.2 GHz. We obtain the CPU performance counter values by perf command.

Table 1. Profiling Results by perf Command

	Circle detection			Kmeans			Face detection		
	OpenMP	pthread	Single	OpenMP	pthread	Single	OpenMP	pthread	Single
CPU Time [s]	261.3	414.2	182.8	60.3	71.8	15.9	175.6	140.2	199.9
#Execution Cycles [M]	305,493	438,507	217,119	72,345	80,104	19,007	206,476	152,051	235,254
#Executed Instructions [M]	104,567	159,691	99,285	49,065	54,771	13,475	116,385	86,133	141,737
IPC	0.34	0.36	0.46	0.68	0.68	0.71	0.56	0.57	0.60
Branches	12,887	19,553	11,967	5,772	5,475	1,352	6,359	4,622	7,556
Branches-Misses	0.44	0.30	0.40	0.78	2.53	2.54	1.76	1.26	1.24
Page-Fault	27,526	26,813	15,206	35,755	111,566	28,578	851,240	504,667	508,497
Context-Switches	2,036	2,684	15,177	754	1,286	1,376	66,703	12,102	18,502
CPU-Migrations	10	95	27	8	94	4	17,441	1,726	176

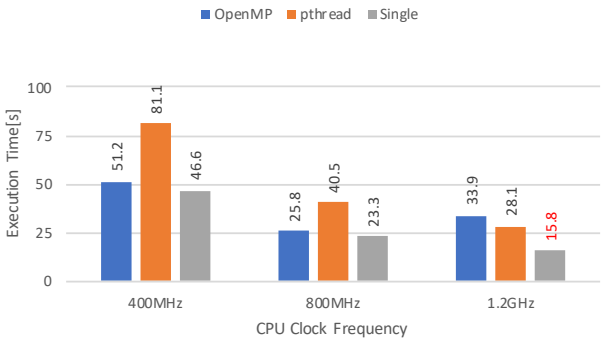


(a) Execution Time

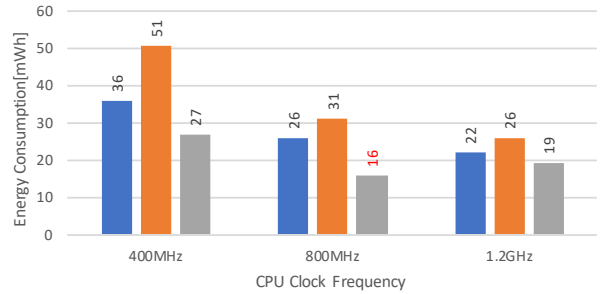


(b) Energy Consumption

Figure 1. Circle Detection Program



(a) Execution Time



(b) Energy Consumption

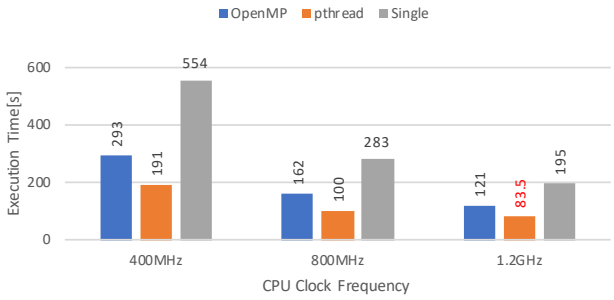
Figure 2. Kmeans Program

Table 1 shows the profiling results. Because Raspberry Pi 3B has 4 CPU cores, CPU time in Table 1 and the actual elapsed time were largely different. In a rough approximation, the elapsed time can be obtained by dividing the CPU time by 4, the number of CPU cores. Table 1 also shows that the number of page faults of the face detection program is much higher than the others. This means that the working set of the face detection is larger than the others.

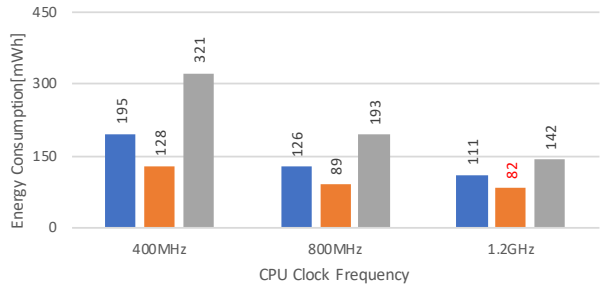
III. Results of Execution Time and Energy Consumption of Single Program

In this experiment, three OpenCV programs were executed on Raspberry Pi with different execution frequencies. Figure 1 shows the execution time and energy consumption of circle detection program. Figure 1 shows that both minimum execution time and minimum energy consumption are obtained at 1.2GHz and OpenMP. The execution time of pthread is shorter than Single-thread, but the impact is relatively small in terms of the number of cores. Energy consumption of pthread is slightly larger than Single-thread. In this experimental environment, power management cannot be applied for each core, and the power consumption of idle cores are also considerable. This is why the energy consumption of Single-thread is not so small.

Figure 2 shows the execution time and energy consumption of Kmeans program. The minimum execution time is obtained



(a) Execution Time



(b) Energy Consumption
Figure 3. Face Detection Program

at 1.2 GHz and Single-thread. On the other hand, the minimum energy consumption is obtained at 800 MHz and Single-thread. Because the given data set is relatively small, it seems that the large benefit of parallelization by OpenMP or pthread cannot be obtained. When we focus on Single-thread execution at Figure 2 (b), the energy consumption of Single-thread at 800MHz is smaller than the 1.2GHz. Because lower execution frequency usually causes lower internal power supply voltage, it seems that the power and energy consumption became small.

Figure 3 shows the execution time and energy consumption of the face detection program. Figure 3 shows that both minimum execution time and minimum energy consumption are obtained at 1.2GHz and pthread.

IV. Results of Execution Time and Energy Consumption of Multiple Programs

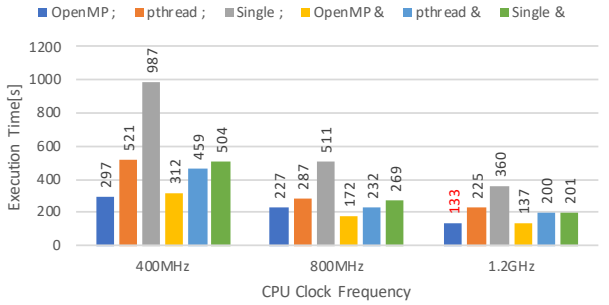
In Section 3, we measured the performance and energy consumption of each single OpenCV program. In this section, we measure the performance and energy consumption of multiple execution. In this experiment, following executions are performed.

- The same program is executed sequentially such as:

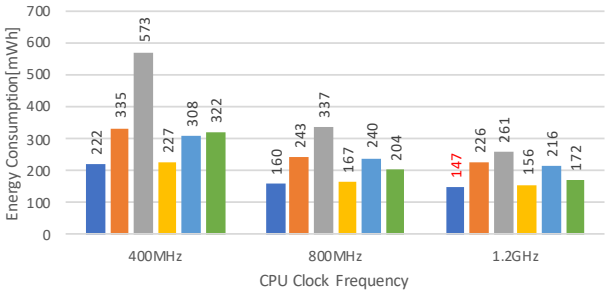
```
$ program; program; program
```

- The multiple identical programs are executed at the same time such as:

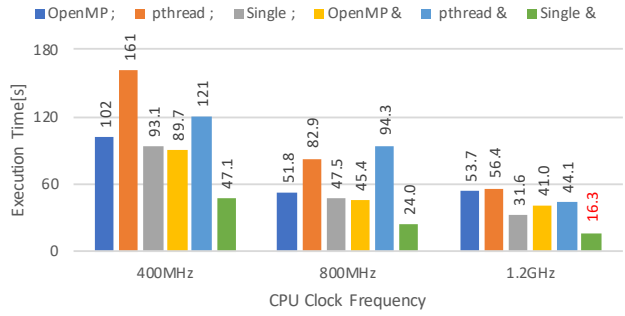
```
$ program1 & program2 & program3
```



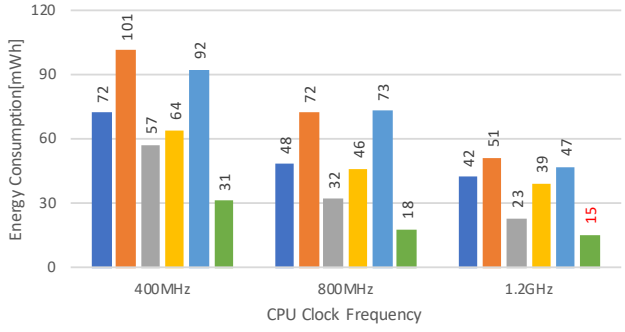
(a) Execution Time



(b) Energy Consumption
Figure 4. Two Circle Programs



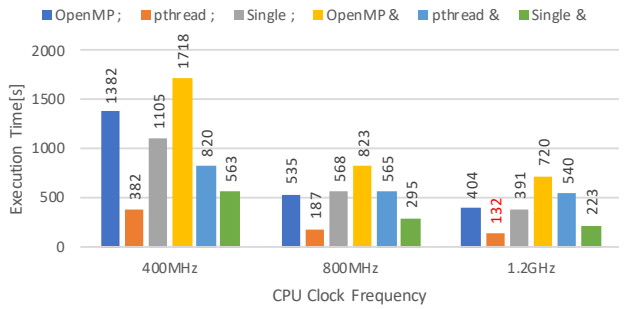
(a) Execution Time



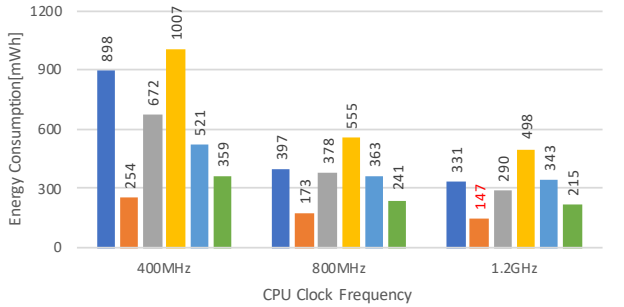
(b) Energy Consumption
Figure 5. Two Kmeans Programs

A. Execution of Two Programs

Figure 4 shows the execution time and energy consumption when two circle detection programs are executed. Figure 4

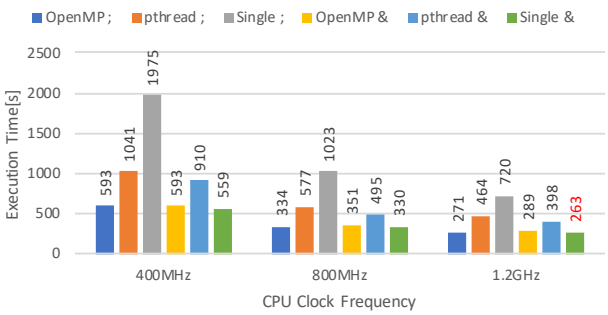


(a) Execution Time

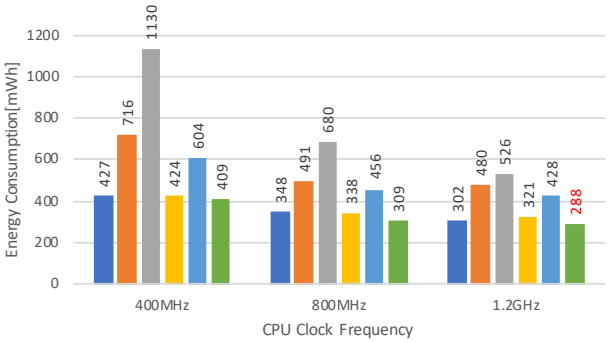


(b) Energy Consumption

Figure 6. Two Face Detection Programs



(a) Execution Time



(b) Energy Consumption

Figure 7. Four Circle Detection Programs

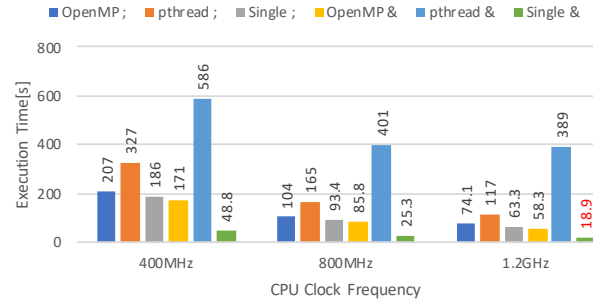
shows that both minimum execution time and minimum energy consumption are obtained at 1.2GHz and OpenMP. This result is the same as the single execution of the same program (circle detection program) as shown in Figure 1. When comparing sequential execution and concurrent execution, sequential execution was slightly better.

Figure 5 shows the execution time and energy consumption when two Kmeans programs are executed. Figure 5 shows that both minimum execution time and minimum energy consumption are obtained at 1.2GHz and Single-thread. This result is the same as the single execution of the same program (Kmeans program) as shown in Figure 2. In the case of Single-thread execution, it is better to execute two programs simultaneously than to execute them sequentially. By executing two programs at the same time, the multiple cores are effectively utilized, and the better results are obtained in the end.

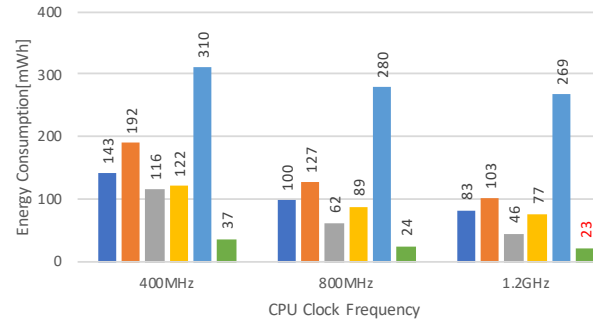
Figure 6 shows the results when two face detection programs are executed. Figure 6 shows that both minimum execution time and minimum energy consumption are obtained at 1.2GHz and pthread.

B. Execution of Four Programs

Figure 7 shows the execution time and energy consumption when four circle detection programs are executed. In case of two circle detection shown in Figure 4, 1.2GHz and OpenMP were the best for both execution time and energy consumption. However, in case of four circle detection shown in Figure 7, 1.2GHz and Single-thread were the best. Since Raspberry Pi that we used in the experiment has four cores, it seems that four Single-thread programs are executed on four cores in parallel.

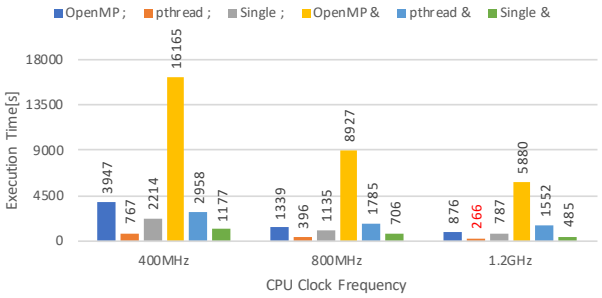


(a) Execution Time

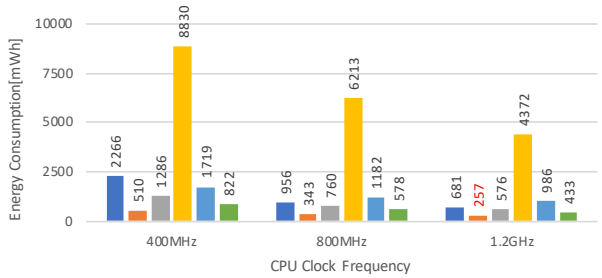


(b) Energy Consumption

Figure 8. Four Kmeans Programs



(a) Execution Time



(b) Energy Consumption

Figure 9. Four Face Detection Programs

Figure 8 shows the execution time and energy consumption when four Kmeans programs are executed. As shown in Figure 8, 1.2GHz and Single-thread were the best for both execution time and energy consumption. Notice that both execution time and energy consumption became worse when four pthread programs were executed.

Figure 9 shows the results when four face detection programs are executed. As shown in Figure 9, 1.2GHz and pthread were the best for both execution time and energy consumption. This is the same results as two face detection programs are executed in Figure 6. On the other hand, the execution time became worse when four OpenMP programs were executed. As mentioned in Section 2.2, the face detection program needs large memory size. Thus, parallel execution of four such programs, which especially execute in data parallel using OpenMP consumes much more memory size. Therefore, the page faults occur frequently.

V. Discussions

OpenCV supports multiple libraries such as OpenMP, pthread, and CUDA. These can be selected when we build the OpenCV. In this experiment, GPU was not used, and multicore parallel execution by OpenMP, multicore parallel execution by pthread, and single thread execution were evaluated. As the results, the parallel libraries with the shortest execution time are different depending on the OpenCV program to be executed or the input data of the program. The suitable parallel libraries are also different depending on the number of and characteristics of the programs executed simultaneously. Furthermore, when executing multiple OpenCV programs, it is not always better to execute them in parallel, and some cases are better to execute them sequentially.

From these results, when we compile the OpenCV application program, it is better to link appropriate OpenCV libraries, which are built with the different parallel libraries. Furthermore, it is better to prepare the appropriate binaries by compiling the application program with the different OpenCV libraries because the characteristics of the execution time and energy consumption are different under the different clock frequency. This means that we have an opportunity to switch the binaries to minimize execution time and energy consumption under the frequency scaling environment.

From the experimental results, the execution time and energy consumption followed the same characteristics for many cases. This means that the minimization of execution time also brings the reduction of energy consumption. Regarding the operating frequency, in most cases, operating at the highest operating frequency (1.2 GHz in this case) was the best in terms of both execution time and energy consumption.

VI. Summery

This paper evaluated the performance and energy consumption of OpenCV programs on Raspberry Pi. We used the following three benchmark applications: circle detection, Kmeans, and face detection. In this paper, we measured the execution time and energy consumption of three types of configurations: parallel execution using OpenMP, parallel execution using pthreads, and single-thread execution. In addition, CPU execution frequency of Raspberry Pi was also changed from 400MHz to 1.2GHz to measure. Experimental results show the best configuration and CPU execution frequency were different to minimize the execution time and energy consumption for each application. These results show the minimization of execution time and energy consumption are achieved by switching the OpenCV library appropriately. Future work includes further experiment and analysis of execution time and energy consumption for OpenCV functions.

Acknowledgment

This research has been partly supported by KIOXIA Corporation (former Toshiba Memory Corporation).

References

[1] G. Bekarro, A. Santokhee, "*Power Consumption of the RaspberryPi: A Comparative Analysis*," International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies, 2016.

[2] C. Cabaccan, F. Reidj, G. Cruz, "*Power Characterization of Raspberry Pi Agricultural Sensor Nodes Using Arduino Based Voltmeter*," *International Conference on Computer and Communication Systems*, 2018.

[3] F. Astudillo-Salinas, D. Barrera-Salamea, "*Minimizing the Power Consumption in Raspberry Pi to Use as a Remote WSN Gateway*," *Latin-American Conference on Communications*, 2016.

[4] F. Kaup, P. Gottschling, D. Hausheer, "*PowerPi: Measuring and Modeling the Power Consumption of the Raspberry Pi*," *Conference on Local Computer Networks*, 2014.