Approach To Sustainable Software – Python Applications

With the growing need for software-driven devices in modern life, the pervasive necessity of energy efficiency is also rising dramatically. This paper analyses the software development process of Python applications in terms of energy consumption. The maturity of energy-efficient programming is very low and the tools to provide data on a program's energy consumption tend to be weak.

Burden, Adam P

Kakkar, Prateek

Abstract

The way software is designed, developed, and deployed can have a major impact on energy consumption. Accordingly, companies should include software in their sustainability efforts. They should articulate a strategy that guides trade-offs and allows for flexibility, review and refine the software development life cycle, and use "sustainable" software to make cloud-based data centers greener.

Acronyms Used

Acronym	Meaning
RAPL	Running Average Power Limit
CPU	Central Processing Unit
GPU	Graphics Processing Unit

Acronym	Definition
RAPL	RAPL is a feature of recent intel processors that provide the energy consumption of the processor. The RAPL formula is designed to measure power consumption of domains (CPU or RAM) in real time.
CPU	CPU is the component of a computer system that controls the interpretation and execution of instructions.
GPU	GPU is a specialized processor originally designed to accelerate graphics rendering. It can process many pieces of data simultaneously, making them useful for machine learning, video editing, and gaming applications.

Table of Contents

Introduction

Brief Overview

As organizations aim to improve their sustainability credentials, they will inevitably consider the impact of hardware and software. This raises the question of what “**sustainable software**”, or “**green software**” is and how to improve software sustainability.

From a sustainability perspective, software and hardware are inseparable. Sometimes the path to improved sustainability will be via more energy-efficient hardware or more-sustainable power sources for existing hardware. In other cases, software changes may reduce emissions or enable extending the life span of older hardware, deferring the sustainability impact of new equipment purchases.

Many programming tactics and tools can reduce the energy used by software. Appropriate tactics depend on the platform and application architecture:

- Algorithms and data structures — Different algorithms for the same task may have very different efficiencies.
- Programming languages — Studies have shown that some programming languages use more energy for the same task than others. For this paper we are focusing on python only.
- Application architecture — For example, effective use of multithreading can make programs more energy-efficient.
- Programming patterns — Certain programming patterns and tactics are much more inefficient than others; for example, tactics that defeat the operating systems energy efficiency measures.

In this paper we intend to showcase the above effects leveraging simple sorting algorithms developed in python. We would cover how to measure energy consumption of software applications; how much energy is consumed by different sorting algorithm and what effects of underlying operating system have on the overall energy consumption.

Paper Outline

The paper is organized into five sections and several subsections. Section 2 gives an overview of the key considerations on measuring energy consumption of software applications. Section 3 describes an approach for calculating the energy consumed by a software application. It provides a brief on the customizations that were developed to measure the energy consumed by an application. Section 4 describes how the data has been analyzed and compared to some standard vehicle exhaust benchmarks. Last section (section 5) provides a brief overview on RAPL (Running Average Power Limit) profiler.

Software Energy Measurement Approach

Measuring Energy Consumed by Software

A known fact, Software does not consume energy or emit any harmful discharge. Common factors of carbon emission are (1) the way software is developed for use and (2) the way software is used. Software runs on hardware, and as the former continues to grow, so does dependence on the machines to make it run. Hence software is also part and parcel of a rapidly growing carbon footprint.

While running applications on-premises or in the cloud, the main consumers of power on a server will be the CPU, the GPU, the memory, and the Tech Stack. Estimating how much each consumes will give an estimate of how much power your server, or your application on a server, consumes.

Key considerations on measuring energy consumption of software applications:

First, a baseline for energy consumption of software is that any hardware will consume power when idle. For absolute numbers, subtract the baseline consumption measurement from the overall consumption measured when running the application. The next section highlights the approach of measuring the energy consumed by software application.

Second, the development of software can be extremely energy intensive. For example, sorting operation takes a significant amount time when the size of the given input increases. This paper aims to compare the execution time of different sorting algorithms – **MergeSort** and **BubbleSort**. The goals are saving time, decreasing power consumption, and saving money.

Brief on Sorting Algorithms

Sorting – Process of rearrangement of a list of elements to the correct order for efficient handling of the elements.

The following sorting algorithms were considered for energy consumption analysis, MergeSort and BubbleSort.

- **MergeSort**: An efficient, general-purpose, and comparison-based sorting algorithm. Works on Divide and Conquer Approach.
- **BubbleSort**: The simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order.

Measuring Energy Consumption

The Energy Profiler records how much energy each server component uses. The Energy Profiler also shows occurrences of system events (sleep) that can affect energy consumption.

Running Average Power Limit (RAPL) is a feature of recent intel processors that provide the energy consumption of the processor and provides interfaces for reporting the accumulated energy consumption of various power domains. (Last section provide an overview of RAPL).

Identified program for energy efficiency calculation using RAPL technology:

- **pyJoules**: Monitors energy consumption of python code, pyJoules support energy consumption calculation of intel cpu and code snippets using the RAPL technology. This module has been leveraged in our application.

Hardware configurations matching the identified profiler (RAPL).

To attain the energy consumption data of software applications, the challenge of identifying the correct system configurations was achieved by collating the below configurations post several experimentations on physical and virtual machines.

1. Programming Language: Python 3.10
2. Operating System: Linux 5.15.0-57-generic
3. CPU and RAM: 2vCore, 8 GB Ram
4. Processor: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz
5. Model Name: Kabylake Server

Extensions to RAPL profiler

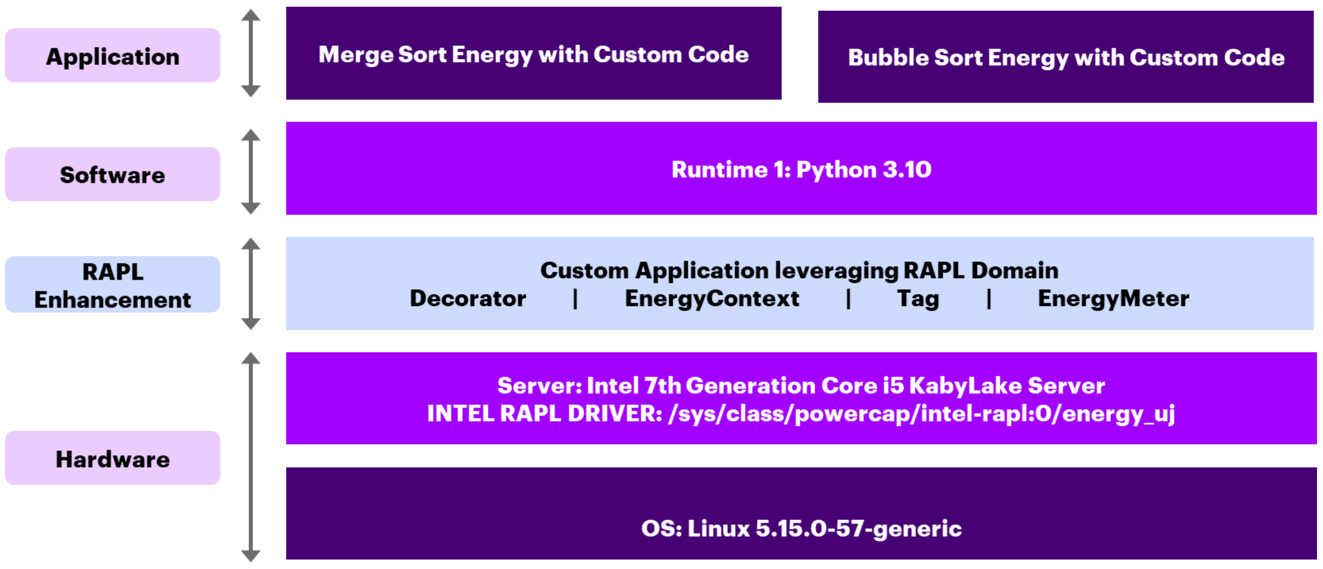
Application
Merge Sort Energy with Custom Code
Bubble Sort Energy with Custom Code
Software
Runtime 1: Python 3.10
RAPL Enhancement
Custom Application leveraging RAPL Domain
Decorator | EnergyContext | Tag | EnergyMeter
Hardware
Server: Intel 7th Generation Core i5 KabyLake Server
INTEL RAPL DRIVER: /sys/class/powercap/intel-rapl:0/energy_uj
OS: Linux 5.15.0-57-generic

Figure 1

As shown in Figure 1 – Custom application "RAPL Enhancement" was built to achieve the following.

1. Collect energy data of the software application.
    a. RAPL Decorator package leverages custom_measure_energy() function for application energy measurement.
2. Helps in removal of Hardware Energy Data.
    a. RAPL Decorator package function custom_measure_hardware_energy() captures CPU and GPU energy measurement.
    b. We need to configure which CPU socket to monitor (applications are deployed on specific CPU). This is achieved through pyRAPL.setup function.
    c. The above data needs to be subtracted from data gathered in Step 1. This ensures only energy data related to application is captured.
3. The last step is to omit the sleep time of a multithreaded application, or the delay time captured in an application.
    a. We leverage the EnergyContext package to add "breakpoint" in specific class/functions.
    b. Start_tag parameter in EnergyContext package with each function name is used to capture energy data for the time when the application is executed.
    c. Energy calculation using ***tagging*** omits the sleep time set in between the two functions.

Sorting Algorithm Data Analysis

Experimental Setup

The following technology configurations are installed to run the sorting applications:

- **Programming Language**: Python 3.10
- **Operating System**: Linux 5.15.0-57-generic
- **CPU and RAM**: 2vCore, 8 GB Ram
- **Processor**: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz
- **Model Name**: Kabylake Server

Scenario overview

Execute MergeSort and BubbleSort on **60,000** elements inheriting the pyJoules library with RAPL Enhancement. The execution flow is shown in Figure 2.

Execution Steps

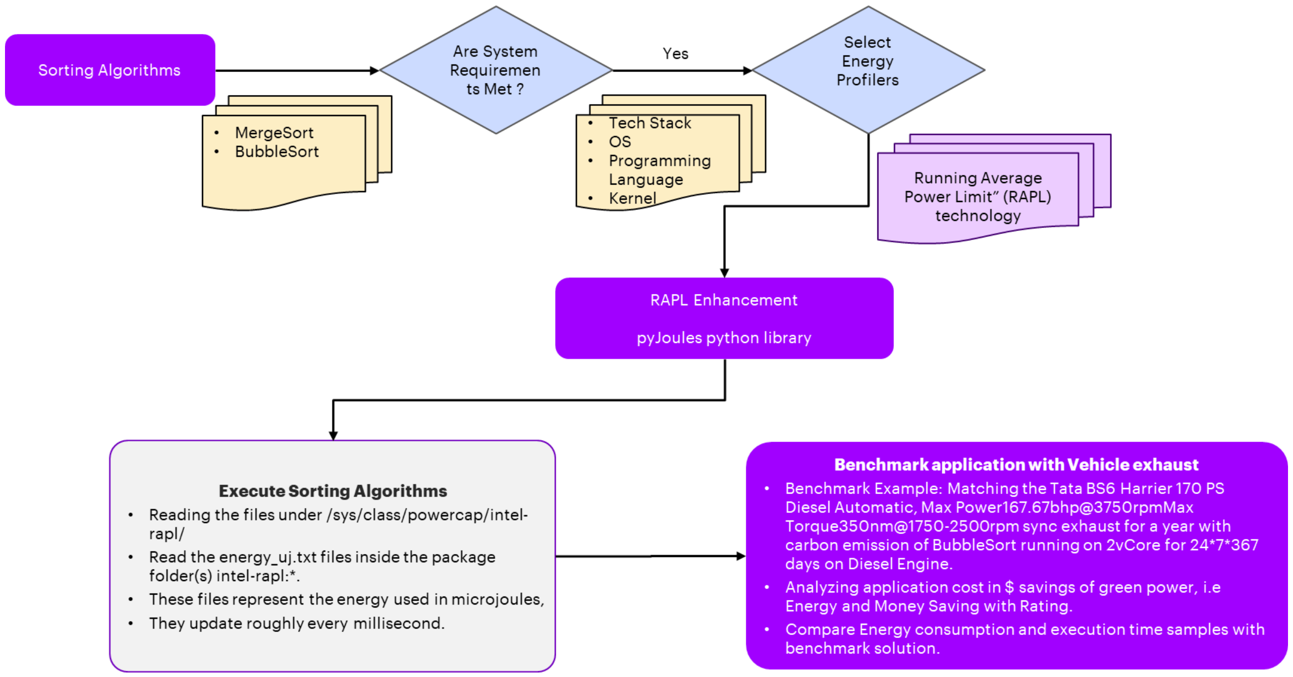
Sorting Algorithms
MergeSort
BubbleSort
Are System Requiremen ts Met ?
Yes
Tech Stack
OS
Programming Language
Kernel
Select Energy Profilers
Running Average Power Limit" (RAPL) technology
RAPL Enhancement
pyJoules python library
Execute Sorting Algorithms
Reading the files under /sys/class/powercap/intel-rapl/
Read the energy_uj.txt files inside the package folder(s) intel-rapl:*.
These files represent the energy used in microjoules,
They update roughly every millisecond.
Benchmark application with Vehicle exhaust
Benchmark Example: Matching the Tata BS6 Harrier 170 PS Diesel Automatic, Max Power167.67bhp@3750rpmMax Torque350nm@1750-2500rpm sync exhaust for a year with carbon emission of BubbleSort running on 2vCore for 24*7*367 days on Diesel Engine.
Analyzing application cost in $ savings of green power, i.e Energy and Money Saving with Rating.
Compare Energy consumption and execution time samples with benchmark solution.

Figure 2

- Below table summarizes the energy consumption of the various sorting algorithm.

Sorting Algorithm	Execution Time in µs	Energy Consumption in µ Joules	Records processed
BubbleSort	536.12	7,357.10	60 K
MergeSort	0.399	4.48	60 K

As shown in the above table, by tunning the algorithms from Bubble Sort to Merge Sort, there is nearly **99.939%** decrease in energy consumption of application.

Benchmark application with Vehicle exhaust

Identification of a benchmark solution.

- Benchmark Example: Tata Nexon XE 170 PS Diesel Automatic, Max Power167.67bhp@3750rpmMax Torque350nm@1750-2500rpm sync
- Matching the Tata Nexon exhaust running on Diesel Engine, for a year with carbon emission along with the MergeSort and BubbleSort application respectively running on 1vCore for 10 hours for 365 days.
- Analyzing application cost in $ savings of green power, i.e., Energy and Money Saving with Rating.
- Compare Energy consumption and execution time samples with benchmark solution.

Conclusions

Results Summary

- BubbleSort Algorithms sorts **60K elements** and consumes **7,357.10 µJ energy in 536.12 µS**.
- MergeSort Algorithms sorts **60K elements** and consumes **4.48 µJ energy in 0.39 µS**.
- Total Energy efficiency gained through change of algorithm is (7357.10 – 4.48) = **7,352.62 µJ**
- Total Energy saved **= 7,352.62 µJ = 2.042394444e-9 kWh**
- Time saved = (536.12 – 0.39) ~ **535.73 µS**
- Time Saving in a day:
  - Assuming 535.73 µS is saved per execution, and algorithm is executed every minute.
  - Hence savings in a day = 535.73 *1440 = **771451.2 µS ~ 0.771 seconds**
- Energy Savings in a day:
  - Assuming 7,352.62 µJ is saved per execution, and algorithm is executed every minute.
  - Energy savings in a day = (7,352.62 * 1440) = **1,05,87,772.8 µJ ~ 10.58 Joule**
  - Energy savings in a year = 10.58 * 365 = **3861.7 Joule = 0.001 kWh**
  - Total number of instances = 2000
  - Total yearly energy savings = 2000 * 3861.7 = **77,23,400 Joules = 2.14 kWh**

Below table evaluates the yearly energy consumption of the benchmark set to Tata Nexon EV exhaust.

Tata Nexon XE	**Specification Data**
Total Distance for a year running 20 days/month	24,500 km
Average Speed	60 km/h
Average Distance per day	100 km
Mileage	16 km/l
Total fuel	1825 L

Tata Nexon XE	**Specification Data**
Per Hour Energy Consumption	**1.9 kWh**

Below table evaluates the yearly energy consumption of the benchmark set to TVS Scooty ES exhaust.

TVS Scooty ES – Petrol	**Specification Data**
Total Distance for a year running 20 days/month	12,250 km
Average Speed	40 km/h
Average Distance per day	50 km
Mileage	16 km/l
Total fuel	720 L
Calculations	1 litre = 0.264172 gallon 720 litre = 194.166 gallon 1 gallon = 1.3×108 Joules 194.166 gallon = **25583312160** Joules
Per Hour Energy Consumption	**0.81 kWh**

To summarize, **MergeSort** is more efficient and consumes less energy than BubbleSort. It also shows that by using a **better sorting algorithm** (MergeSort), the energy saved is approximately equivalent to **1.5 hours** of Tata Nexon XE power exhaust / **2.6 hours** of TVS Scooty ES exhaust.

Brief on RAPL

Energy consumption leveraging RAPL.

Running Average Power Limit (RAPL) is a feature of recent intel processors that provide the energy consumption of the processor and provides interfaces for reporting the accumulated energy consumption of various power domains. Applications are deployed on CORE domain.

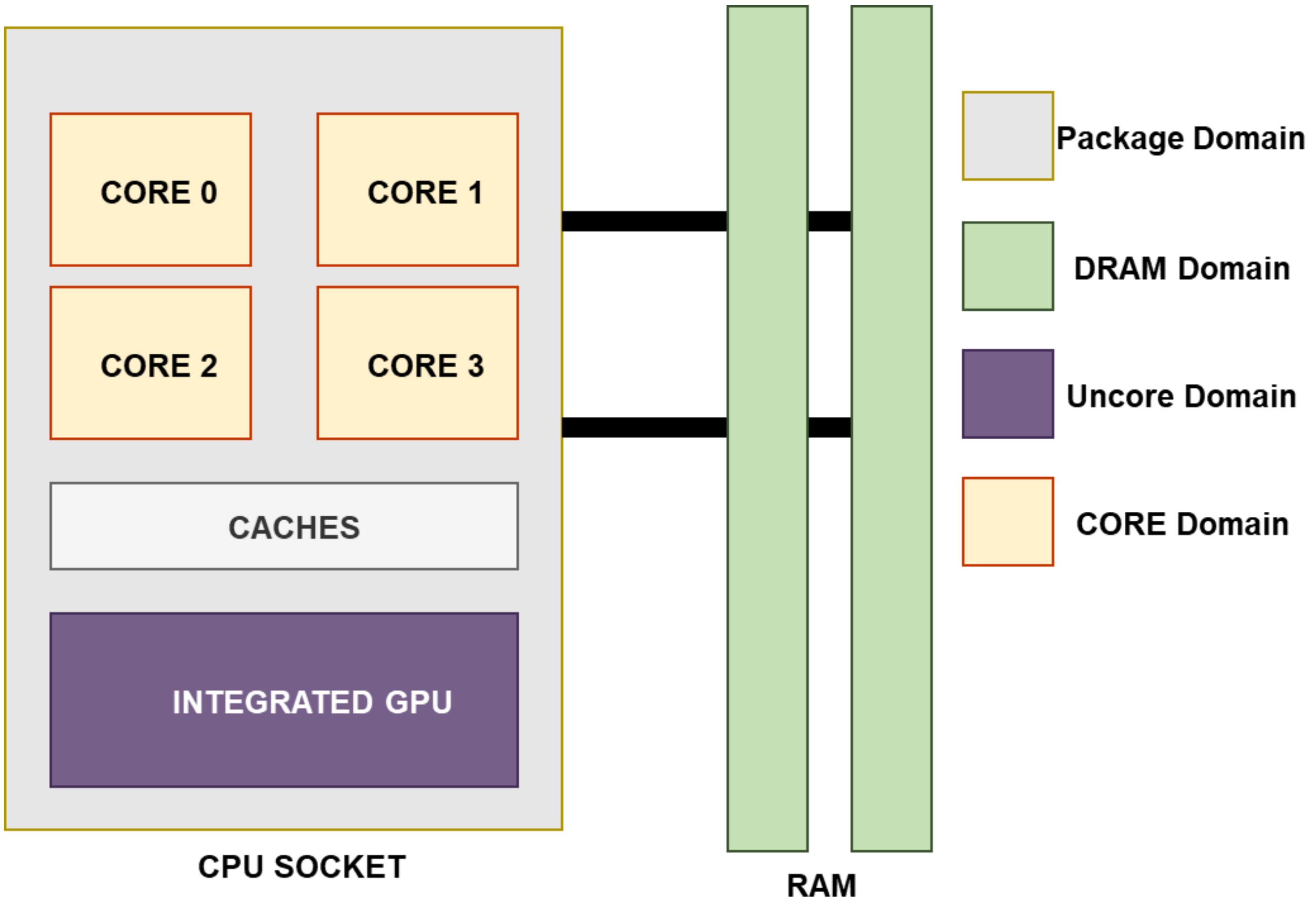

Figure 3

Identified programs for energy efficiency calculation using RAPL technology:

- **pyJoules**: Monitors energy consumption of python code, pyJoules support energy consumption calculation of intel cpu and code snippets using the RAPL technology.
- **Jouleit**: Script to monitor energy consumption for any program.

Power Measurement

RAPL

- Power usage is calculated via the RAPL (Running Average Power Limit) interfaces found on Intel processors.
- The Energy Status register (MSR_PKG_ENERGY_STATUS) allows for power measurement.
- Multiple ways to access the RAPL interface data, namely:
  - Using perf_event interface
  - Reading the underlying MSR
  - Reading the files under `/sys/class/powercap/intel-rapl/`
    - Read the `energy_uj.txt` files inside the package folder(s) `intel-rapl:*`.
    - These files represent the energy used in microjoules, and they update roughly every millisecond.
- INIT: Run rapl executable file (main) appended with application
- Calculate Energy using pow() function in MSR_PKG_ENERGY_STATUS package
  - Rapl_Before()
  - Software Application executed.
  - Rapl_After()
- Result: (Rapl_After – Rapl_Before), The recorded energy consumption is stored in a .csv file