Ramdayal

Task : power consumption of an single application

problem statement

Specific power consumption of an application

Tech-stack and Libraries

In this task i was used python language as a programming language and psutil python library for fetching information.

```
------about psutil library------
```

-> psutil is a cross-platform library for retrieving information on system utilization (CPU, memory, disks, network, sensors) and on system uptime. It is written in Python and is primarily used for system monitoring, profiling, and limiting process resources and system uptime. The library provides a wide range of functionalities such as getting system uptime, system boot time, system memory usage, process information, system temperature, and many more.

-> it can be used in conjunction with other tools to measure the energy consumption of an application.

-> One approach is to use psutil to measure the CPU usage of a single application over a certain period of time. Then, using the measured CPU usage and the known power consumption of the system, we can estimate the energy consumption of the application.

-> psutil to get the process object of an application and then use the cpu_percent() function to get the CPU usage of that process. we can then multiply the CPU usage by the known power consumption of the system to get an estimate of the energy consumption of the application.

Source-Code explanation

This source code is a Python script that uses the psutil library to measure the power consumption of a single application on a system. The script defines several functions to accomplish this task:

(1) The get_process_pids() function takes a process name as an input and returns a list of process IDs that are related to that single application. It does this by iterating over all running processes on the system and matching the process name to the input name. If a match is found, the corresponding process ID is appended to the list.

(2) The Power_consumption() function takes a list of process IDs as input and calculates the power consumption of those processes, as well as the total power consumption of all running processes. It does this by iterating over all running processes and using the psutil library to access information about the process, including its CPU times. It then calculates the duration of the process and the energy consumed by the process based on that duration and the process's CPU usage. The energy consumed by the process is added to the total energy consumed, and if the process ID is in the input list, it's added to the energy consumed by the single application.

(3) The power_consumption_measurement() function is the main function of the script. It prompts the user to input the name of a single application and calls the get_process_pids() function to get a list of process IDs related to that application. It then calls the Power_consumption() function with that list of process IDs to calculate the power consumption of the single application and the total power consumption. It then calculates the percentage of power consumed by the single application and prints the results.

Finally, the power_consumption_measurement() function is called to run the script.

In [16]:

```
#psutil (process and system utilities) library
import psutil


#get_process_pids function return the list of process id which is related to particular sin
def get_process_pids(process_name):
    #process or single application file extension
    extension=".exe"
    #add the extension at the end off application name
    process=process_name+extension
    #Take a list which store the all processes of single application which are currently ru
    List=[]

    #psutil.process_iter function iterate over all running processes on the system.
    for proc in psutil.process_iter():
        #exception handling using try and except
        try:
            #iterate all process name (currently in running state) and matching with the in
            if proc.name() == process:
                #when we get desired process just append in to the List
                List.append(proc.pid)

        #any abnormal situation occure just pass
        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
            pass
    #return the list of process id's
    return List



#power consumption function measure the power consumed by single application and also total
#running application
def Power_consumption(pids_list):
    #varible of power_consumption by process intialize with 0
    single_Application_power_consumption = 0
    # Initialize a variable to store the total energy consumed
    total_power_consumed = 0

    # Iterate over all running processes
    for proc in psutil.process_iter():
        #exception handling using try and except
        try:
            # Get the process information
            process = psutil.Process(proc.pid)
            info = process.as_dict(attrs=['pid', 'name', 'cpu_times'])
            # Get the CPU times for the process
            cpu_times = info['cpu_times']
            # Calculate the duration of the process
            duration = cpu_times.user + cpu_times.system
            # Calculate the energy consumed by the process
            energy = duration * proc.cpu_percent() / 100
            # Add the energy consumed by this process to the total energy
            total_power_consumed += energy

            #if the current pid is related to single application
            if(proc.pid in pids_list):
                single_Application_power_consumption+=energy

        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
            pass
    # Return the ans list which contains both value single ans well as total energy consump
    ans_List = []
    ans_List.append(single_Application_power_consumption)
    ans_List.append(total_power_consumed)

    #return ans list
    return ans_List




#power_consumption function of single application
def power_consumption_measurement():
    #take the input as a single application name for measuring the power consumption
    process_name = input('Type the name of the single application for measuring power consu
    print("Application name: ", process_name)

    #call the get_process_pid function
    pids = get_process_pids(process_name)


    #call the function for power consumptio with the help of psutil python library
    power_consumption = Power_consumption(pids)

    #single application power consumption
    print("Power consumption of single application : ", power_consumption[0]);

    #calculate total power consumption
    print("Total Power Consumption : ",power_consumption[1])

    power_consumption_ratio = ( power_consumption[0]/power_consumption[1])*100
    print("%power consumed by single application",power_consumption_ratio)



#call the power_consumption
power_consumption_measurement()
```

```
Type the name of the single application for measuring power consumption: Gra
mmarly.Desktop
Application name:  Grammarly.Desktop
Power consumption of single application :  3.3451875
Total Power Consumption :  11043542.239078127
%power consumed by single application 3.029089242908753e-05
```

Conclusion

Languages ,Frameworks ans libraries

There are several programming languages, frameworks, and libraries that can be used to measure the power consumption of a single application:

PowerAPI (Java): PowerAPI is a library for measuring the power consumption of applications written in Java. It uses the Java Virtual Machine Profiling Interface (JVMPI) to measure the power consumption of an application by collecting data on the application's CPU usage, memory usage, and other system resources.

Psutil (Python): Psutil is a cross-platform library for process and system utilities in Python. It can be used to measure the power consumption of a single application by accessing information about the process, including its CPU times, and calculating the duration of the process and the energy consumed by the process based on that duration and the process's CPU usage.

PowerTOP (C++): PowerTOP is a tool for measuring the power consumption of applications written in C++. It uses the Linux kernel's performance monitoring infrastructure to gather data on the power usage of running processes and provides a detailed analysis of the power consumption of an application.

It's worth noting that the accuracy and functionality of these libraries and tools can vary depending on the specific device and operating system they are used on.

Tools

In addition to the programming languages, frameworks, and libraries mentioned earlier, there are also several tools that can be used to measure the power consumption of a single application:

Intel Power Gadget: Intel Power Gadget is a tool developed by Intel that can be used to measure the power consumption of an application on Windows and macOS. It provides detailed information on the power usage of the CPU, GPU, memory, and other system components, and can be used to create custom power usage profiles for different applications.

JPower: JPower is a Java-based tool for measuring the power consumption of applications on Linux-based systems. It uses the Linux kernel's performance monitoring infrastructure to gather data on the power usage of running processes and provides a detailed analysis of the power consumption of an application.

PowerTOP: PowerTOP is a tool for measuring the power consumption of applications on Linux-based systems. It uses the Linux kernel's performance monitoring infrastructure to gather data on the power usage of running processes and provides a detailed analysis of the power consumption of an application.

PowerDebug: PowerDebug is a tool that can be used to measure the power consumption of applications on Linux-based systems. It uses the Linux kernel's performance monitoring infrastructure to gather data on the power usage of running processes and provides a detailed analysis of the power consumption of an application.

Energy Logger: Energy Logger is a tool that can be used to measure the power consumption of applications on Linux-based systems. It uses the Linux kernel's performance monitoring infrastructure to gather data on the power usage of running processes and provides a detailed analysis of the power consumption of an application.

These tools can be used to measure the power consumption of a single application on a specific device, and depending on the device and its capabilities, the results can vary.