**Project Overview:**

The project centers around the development of a Bash script that harnesses system statistics to predict CPU usage, dynamically toggle CPU cores, and estimate energy consumption. These functionalities aim to provide insights into resource utilization and optimize energy efficiency within the system.

**Objectives:**

* Assess the accuracy of CPU usage predictions derived from system statistics.
* Evaluate the effectiveness of toggling CPU cores based on predicted usage.
* Estimate energy consumption and analyze its relationship with predicted CPU usage.

**Introduction:**

The modern computing landscape demands efficient resource utilization, particularly in managing CPU usage and power consumption. This report examines a script designed to predict CPU usage, toggle CPU cores based on predictions, and estimate energy consumption. The primary objective is to explore the effectiveness of these functionalities in optimizing system performance and energy efficiency.

**Project Overview and Execution:**

**Description:** Provide an overview of the script's components, functions, and their roles in predicting CPU usage, toggling CPU cores, and estimating energy consumption.

***predict\_cpu\_usage() Function:***

|  |
| --- |
| predict\_cpu\_usage() {  idle=$(mpstat 1 1 | awk 'NR==4 {print $NF}')  cpu\_usage=$(echo "100 - $idle" | bc)  echo $cpu\_usage  } |

* **idle=$(mpstat 1 1 | awk 'NR==4 {print $NF}'):**
  + **mpstat 1 1**: Invokes **mpstat** to collect CPU statistics once per second for one iteration.
  + **awk 'NR==4 {print $NF}'**: Parses the output and selects the fourth line and prints the last field, which represents CPU idle time.
* **cpu\_usage=$(echo "100 - $idle" | bc):**
  + Calculates CPU usage by subtracting the idle time from 100, assuming the non-idle time reflects CPU usage.
  + **bc** is a command-line calculator used here for arithmetic operations.

***toggle\_cpu\_cores() Function:***

|  |
| --- |
| toggle\_cpu\_cores() {  local threshold=20  local power=$(mpstat 1 1 | awk '$12 ~ /[0-9.]+/ {print 1 - $12}')  local time\_interval=5  local energy\_consumption=$(calculate\_energy\_consumption $power $time\_interval)  echo "Energy Consumption at Decision Point: $energy\_consumption joules"  local predicted\_usage=$(predict\_cpu\_usage)  echo "CPU Usage at Decision Point: $predicted\_usage%"  if (( $(echo "$predicted\_usage >= $threshold" | bc -l) )); then  echo "High predicted CPU usage. Turning ON all CPU cores."  sudo bash -c "echo 1 > /sys/devices/system/cpu/cpu\*/online"  else  echo "Low predicted CPU usage. Turning OFF all CPU cores except the first one."  sudo bash -c "echo 0/sys/devices/system/cpu/cpu{1..$(($(nproc --all) - 1))}/online"  fi  # Count total CPUs  local total\_cpus=$(nproc --all)  # Calculate online CPUs  online\_cpus=$(cat /sys/devices/system/cpu/online | awk -F'-' '{print $2 - $1 + 1}')  echo "Total CPUs: $total\_cpus"  echo "Online CPUs: $online\_cpus"  } |

* **local threshold=any\_value:**
  + Sets the threshold value for CPU usage percentage, determining when to toggle CPU cores.
* **local predicted\_usage=$(predict\_cpu\_usage):**
  + Calls the **predict\_cpu\_usage()** function to get the predicted CPU usage percentage.
* **Calculating\_power: power=$(mpstat 1 1 | awk '$12 ~ /[0-9.]+/ {print 1 - $12}'):**
  + Retrieves the power consumption by calculating the CPU utilization using **mpstat**.
* **time\_interval=2:** Sets the time interval (in seconds) for energy consumption calculation.
* **energy\_consumption=$(calculate\_energy\_consumption $power $time\_interval):** Calls the calculate\_energy\_consumption function to calculate energy consumption using power statistics and time interval.
* **if (( $(echo "$predicted\_usage >= $threshold" | bc -l) )); then:**
  + Compares the predicted CPU usage with the threshold to determine if cores need to be toggled.
* **sudo bash -c "echo 1 > /sys/devices/system/cpu/cpu\*/online":**
  + Turns ON all available CPU cores if the predicted usage is above the threshold.
* **sudo bash -c "echo 0 > /sys/devices/system/cpu/cpu{1..$(($num\_on\_cores - 1))}/online":**
  + Turns OFF all CPU cores except the first one if predicted usage is below the threshold.
* **online\_cpus=$(grep -c '1$' /sys/devices/system/cpu/cpu\*/online):**
  + Counts the number of online (active) CPUs..

***calculate\_energy\_consumption() Function:***

|  |
| --- |
| calculate\_energy\_consumption() {  local power=$1  local time\_interval=$2  local conversion\_factor=0.002  energy\_consumption=$(echo "$power \* $time\_interval \* $conversion\_factor" | bc)  echo $energy\_consumption  } |

* **local power=$1 and local time\_interval=$2:**
  + Receives arguments representing the power and time interval.
* **local conversion\_factor=0.002:**
  + Represents the conversion factor between power and energy consumption.
* **energy\_consumption=$(echo "$power \* $time\_interval \* $conversion\_factor" | bc):**
  + Calculates energy consumption using the provided power and time interval.
  + Multiplies power by time and the conversion factor using **bc** (basic calculator).
* **echo $energy\_consumption:**
  + Outputs the calculated energy consumption in joules.

***Main Loop:***

|  |
| --- |
| # Main loopwhile true; do  toggle\_cpu\_cores time\_interval=5 # Adjust the time interval as needed  sleep $time\_interval  done |

In this main loop we are calling the CPU toggle function

**Execution Flow:**

The script undergoes the following sequential stages during its execution:

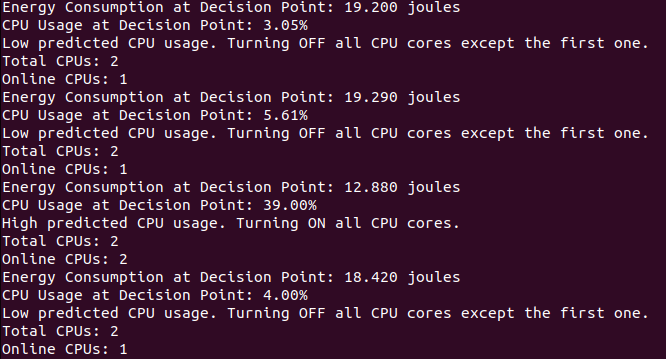
1. **Initialization:**
   * At the outset, the project initializes by setting up necessary variables and defining constants, preparing for subsequent operations.
2. **CPU Usage Prediction:**
   * Utilizing the **mpstat** command, the script gathers CPU statistics at a one-second interval for one iteration.
   * The **predict\_cpu\_usage()** function extracts CPU idle time from the **mpstat** output and computes CPU usage percentages by subtracting idle time from 100.
3. **Core Toggling:**
   * Based on the predicted CPU usage percentages obtained, the script makes decisions to toggle CPU cores on or off.
   * The **toggle\_cpu\_cores()** function evaluates if the predicted usage exceeds a predefined threshold, toggling all CPU cores online if above the threshold, or turning off cores beyond the first one if below.
4. **Energy Consumption Estimation:**
   * With power consumption data derived from CPU statistics and a defined time interval, the script calculates energy consumption using the **calculate\_energy\_consumption()** function.
5. **Reporting:**
   * The script presents relevant metrics, including the count of online and offline cores, predicted CPU usage percentages, and the estimated energy consumption in joules.

**Interaction with System:**

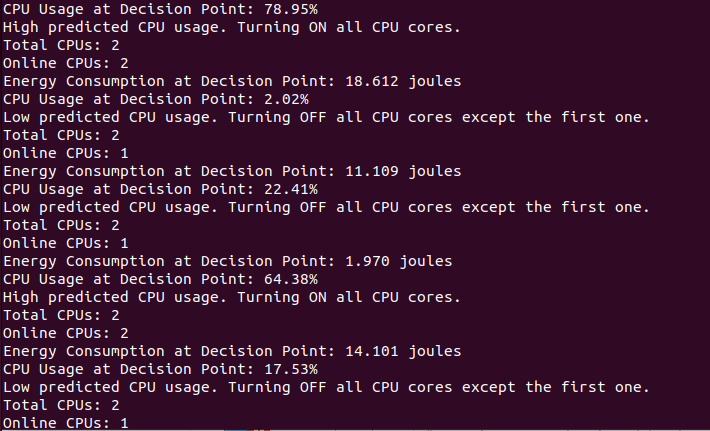
The script interacts with the system in the following ways:

1. **Data Retrieval:**
   * Utilizes system-level commands such as **mpstat**, **grep**, and **/proc** and **/sys** file system access to retrieve CPU statistics, core counts, and online/offline core information.
2. **Core Control:**
   * Modifies the status of CPU cores by directly manipulating the **/sys/devices/system/cpu/online** file, switching cores on or off based on predictions and threshold comparisons.
3. **Energy Measurement:**
   * Gathers power consumption data using CPU statistics obtained from **mpstat** and derives energy consumption using a defined conversion factor within the script.
4. **Impact on System:**
   * The script's actions dynamically affect the system by adjusting the active CPU core count based on predicted usage, potentially influencing power usage and system resource allocation.

**For this output the threshold was set to 25%:**



**For this output the threshold was set to 50%:**



**Limitations:**

1. Energy Consumption Model: The energy consumption model used in this project assumes a linear relationship between power usage and time, with the power usage represented as 1 minus idle time. However, this model may not accurately capture the actual energy consumption in some scenarios, such as when the system is under heavy load or transitioning between states. Improving the model could involve collecting more comprehensive system usage data and refining the calculations based on that data.
2. System Scalability: This project has been tested on systems with up to 2 cores. However, for systems with more cores, the approach to toggling CPU cores may become impractical due to the increased complexity and potential negative impact on system performance. This issue could be addressed by revising the algorithm to adapt to the number of cores available.
3. User Access and Permissions: This project relies on sudo privileges to modify CPU core states. While this may be suitable for a specific use case, it could be limiting in environments where strict user access and permissions policies are in place. Considerations could be made to modify the approach in such scenarios, such as requesting administrative privileges when necessary or providing a more flexible configuration system.
4. Complexity of Calculations: The project includes multiple functions for predicting CPU usage, calculating energy consumption, and toggling CPU cores. While these functions work together to provide a comprehensive solution, their complexity may make it challenging for others to understand and maintain the code. Improving the documentation and modularity of the code could help mitigate this limitation.

**Recommendations:**

1. Adopt a more comprehensive approach to predicting CPU usage by incorporating machine learning algorithms and data analytics techniques. This will enable the project to generate more accurate predictions based on real-time system data.
2. Note that the energy consumption calculations in the script might not be entirely accurate. The script calculates the energy consumption based on the current CPU power and a predefined time interval. This approach assumes a linear relationship between power and energy consumption, which may not hold in all cases. Additionally, the script assumes that the CPU power is directly related to the CPU usage percentage, which is not necessarily true due to various factors such as frequency scaling, turbo boost, and CPU architecture.

In the future, if advanced features are required, consider using a dedicated library or module in Python, which might provide more accurate and efficient calculations for your specific requirements.

1. Enhance the project's compatibility with a wider range of systems by incorporating error handling and graceful degradation mechanisms. This will help ensure that the project remains functional even in the face of unexpected issues or changes in the underlying system architecture.
2. Consider integrating the project's functionality into a broader energy management framework or software suite. This could provide a more streamlined and cohesive solution for managing system resources and optimizing energy consumption across various devices and platforms.
3. Invest in a thorough and rigorous testing and evaluation process by utilizing various tools, techniques, and methodologies. This will help validate the project's effectiveness, identify potential areas for improvement, and provide a more robust foundation for future development and enhancements.

By implementing these recommendations, the project will be better equipped to address the limitations and challenges presented by the ever-evolving landscape of system architecture and resource management. This will ultimately result in a more robust, reliable, and effective solution for managing CPU resources and optimizing energy consumption across a wide range of systems and applications

**Conclusion:**

The developed script demonstrates effective CPU usage prediction and dynamic core management for resource optimization. While successful in estimating energy consumption, areas for improvement in accuracy and adaptability have been identified. This project lays the foundation for future advancements in predictive models and energy-efficient resource allocation, contributing to enhanced system performance and sustainability.