Carbon measurement and energy attribution for processes and hardware devices in Linux

Aditya Manglik
ETH Zürich, Switzerland

LinkedIn: linkedin.com/in/adityamanglik/
Email: amangli@ethz.ch

FOSDEM-2024
3 February, 2024
Graduate student at ETH Zürich, Switzerland
Graduate student at ETH Zürich, Switzerland

Research at the intersection of computer architecture and operating systems
Outline

Background
Problem
Goal
Current Tools
  PowerTOP
System Design
End Product
Conclusion
Energy sources in computing systems:
Direct: DC input / USB / Ethernet
Energy sources in computing systems:

- **Direct**: DC input / USB / Ethernet
- **Battery**
Energy sources in computing systems:
- Direct: DC input / USB / Ethernet
- Battery
- Energy harvesting
Background

- Energy sources in computing systems:
  - Direct: DC input / USB / Ethernet
  - Battery
  - Energy harvesting

- We want to use the maximum minimum amount of energy to perform computation
Background

- Energy sources in computing systems:
  Direct: DC input / USB / Ethernet
  Battery
  Energy harvesting

- We want to use the maximum minimum amount of energy to perform computation

- Battery capacity is a major design constraint and UX aspect for any consumer device: cellphones and AR/VR headsets
Calculating Energy Consumption of Software

Energy Consumption = Power \times \text{Latency}
Calculating Energy Consumption of Software

Energy Consumption = \text{Power} \times \text{Latency}

\text{Power} \text{ is determined by hardware}
Calculating Energy Consumption of Software

Energy Consumption = \textit{Power} \times \textit{Latency}

\textit{Power} is determined by hardware

\textit{Latency} is determined by software
Calculating Energy Consumption of Software

Energy Consumption = \text{Power} \times \text{Latency}

\text{Power} \text{ is determined by hardware}

\text{Latency} \text{ is determined by software}

Programmers often optimize \text{latency} using well-established tools (e.g., perf) and metrics (e.g., CPU clock cycles)
Calculating Energy Consumption of Software

Energy Consumption = Power \times \text{Latency}

\text{Power} is determined by hardware

\text{Latency} is determined by software

Programmers often optimize \text{latency} using well-established tools (e.g., perf) and metrics (e.g., CPU clock cycles)

Question: \text{Tools to measure application’s energy?}
Calculating Energy Consumption of Software

Energy Consumption = Power × Latency
Calculating Energy Consumption of Software

Energy Consumption = Power × Latency

Power is reported by the CPU (e.g., RAPL interface)
Calculating Energy Consumption of Software

Energy Consumption = Power \times \text{Latency}

Power is reported by the CPU (e.g., RAPL interface)

Example: CPU \approx 15 \text{ W}
Calculating Energy Consumption of Software

Energy Consumption = \text{Power} \times \text{Latency}

\text{Power} \text{ is reported by the CPU (e.g., RAPL interface)}

Example: CPU \approx 15 \text{ W}

\text{Latency} \text{ is determined by software}
Calculating Energy Consumption of Software

Energy Consumption = \textbf{Power} \times \textbf{Latency}

\textbf{Power} is reported by the CPU (e.g., RAPL interface)

Example: CPU \approx 15 \text{ W}

\textbf{Latency} is determined by software

Example: Application X \approx 5 \text{ ms}
Calculating Energy Consumption of Software

Energy Consumption = \textit{Power} \times \textit{Latency}

\textit{Power} is reported by the CPU (e.g., RAPL interface)

Example: CPU \approx 15 \text{ W}

\textit{Latency} is determined by software

Example: Application \textit{X} \approx 5 \text{ ms}

Energy Consumption = \textit{15 W} \times 5 \text{ ms} = 75 \text{ mJ}
Calculating Energy Consumption of Software

Energy Consumption = Power × Latency

Power is reported by the CPU (e.g., RAPL interface)

Example: CPU ≈ 15 W

Latency is determined by software

Example: Application X ≈ 5 ms

Energy Consumption = 15 W × 5 ms = 75 mJ

Problem: Does not reflect ground reality!
Calculation Model

- The model assumes linear power draw
Calculation Model

- The model assumes linear power draw

**Figure:** CPU Power Consumption over time
Calculation Model

▶ The model assumes linear power draw

Figure: CPU Power Consumption over time

▶ **Limitation 1**: Power consumption (on y-axis) is not linear over time (on x-axis)
Platform-specific interfaces: RAPL is available only on Intel
Ground Truth

- Platform-specific interfaces: RAPL is available only on Intel
- AMD and ARM have different interfaces
Platform-specific interfaces: RAPL is available only on Intel
AMD and ARM have different interfaces

Limitation 2: We do not have uniform interfaces and formats needed to measure power reliably across different platforms
Calculation Model

- The model focuses on the CPU
Calculation Model

- The model focuses on the CPU
- **Limitation 3**: What about devices like memory (DRAM), screen, and network cards?
The model focuses on the CPU

Limitation 3: What about devices like memory (DRAM), screen, and network cards?

Experiments are contrary to assumptions, findings similar to Google [1]

Problem Summary

▶ We are *inaccurately* calculating only *a fraction* of the system’s actual energy consumption!
Problem Summary

▶ We are *inaccurately* calculating only *a fraction* of the system’s actual energy consumption!

▶ Summary: We cannot improve what we cannot measure.
Goal

Develop a framework to accurately and reliably measure the energy consumption of the applications on Linux.
Goal

Develop a framework to *accurately and reliably* measure the energy consumption of the applications on Linux.

Report the statistics to the
Goal

Develop a framework to accurately and reliably measure the energy consumption of the applications on Linux.

Report the statistics to the
- End-users: In an easy-to-understand and useful format
  - Programmers: Via APIs that improve programmer actionability
  - System Designers: To enable iterating over low-energy designs
Develop a framework to accurately and reliably measure the energy consumption of the applications on Linux.

Report the statistics to the
- **End-users**: In an easy-to-understand and useful format
- **Programmers**: Via APIs that improve programmer actionability
Develop a framework to accurately and reliably measure the energy consumption of the applications on Linux.

Report the statistics to the
- **End-users**: In an easy-to-understand and useful format
- **Programmers**: Via APIs that improve programmer actionability
- **System Designers**: To enable iterating over low-energy designs
Goal

- Framework = Models and Tools
Goal

Framework = Models and Tools

Power models = How we reason about and estimate a device’s power draw over time
Goal

- **Framework** = Models and Tools
- **Power models** = How we reason about and estimate a device’s power draw over time
- **Power models** are often not available or poorly understood for many devices, e.g., DRAM
Goal

- **Framework** = Models and Tools
- **Power models** = How we reason about and estimate a device’s power draw over time
- **Power models** are often not available or poorly understood for many devices, e.g., DRAM
- **Tools** can be built to accurately calculate power using the models, e.g., nvidia-smi
Goal

- **Framework** = Models and Tools
- **Power models** = How we reason about and estimate a device’s power draw over time
- **Power models** are often not available or poorly understood for many devices, e.g., DRAM
- **Tools** can be built to accurately calculate power using the models, e.g., nvidia-smi
- **Takeaway:** We need accurate models and reliable tools to calculate energy consumption
Outline

Background
Problem
Goal
Current Tools
PowerTOP
System Design
End Product
Conclusion
PowerTOP

Summary: 1541.8 wakups/second, 42.9 GPU ops/seconds, 0.0 VFS ops/sec and 18.9% CPU use

<table>
<thead>
<tr>
<th>Power est.</th>
<th>Usage</th>
<th>Events/s</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.45 W</td>
<td>0.9 ms/s</td>
<td>315.3</td>
<td>Device</td>
<td>nic:virbr0</td>
</tr>
<tr>
<td>1.45 W</td>
<td>38.7 ms/s</td>
<td>54.7%</td>
<td>Process</td>
<td>/usr/bin/gnome-shell</td>
</tr>
<tr>
<td>353 mW</td>
<td>292 mW</td>
<td>10.7 ms/s</td>
<td>Device</td>
<td>Display backlight</td>
</tr>
<tr>
<td>200 mW</td>
<td>7.4 ms/s</td>
<td>57.6</td>
<td>Process</td>
<td>/usr/libexec/Xorg vt4 -displayfd 3</td>
</tr>
<tr>
<td>110 mW</td>
<td>4.9 pkts/s</td>
<td>92.4</td>
<td>Device</td>
<td>Network interface: wlp2s0 (iwlwifi)</td>
</tr>
<tr>
<td>7.31 mW</td>
<td>1.3 ms/s</td>
<td>92.4</td>
<td>Process</td>
<td>/usr/libexec/at-spi2-registryd --user</td>
</tr>
<tr>
<td>0 mW</td>
<td>8.7 ms/s</td>
<td>62.0</td>
<td>Process</td>
<td>/opt/google/chrome/chrome --type=ui</td>
</tr>
<tr>
<td>0 mW</td>
<td>5.4 ms/s</td>
<td>385.4</td>
<td>Interrupt</td>
<td>PS/2 Touchpad / Keyboard / Mouse</td>
</tr>
<tr>
<td>0 mW</td>
<td>4.9 ms/s</td>
<td>79.0</td>
<td>Process</td>
<td>/opt/google/chrome/chrome</td>
</tr>
<tr>
<td>0 mW</td>
<td>4.4 ms/s</td>
<td>2.5</td>
<td>Process</td>
<td>/usr/bin/python /usr/bin/powerline</td>
</tr>
<tr>
<td>0 mW</td>
<td>4.3 ms/s</td>
<td>163.0</td>
<td>Process</td>
<td>powertop</td>
</tr>
<tr>
<td>0 mW</td>
<td>3.6 ms/s</td>
<td>18.6</td>
<td>Process</td>
<td>gnome-shell --mode=xdg --wayland--no-modeswitch</td>
</tr>
</tbody>
</table>
It is possible to use Powertop to view the "power estimate" of a process/device/interrupt/timer.
It is possible to use Powertop to view the "power estimate" of a process/device/interrupt/timer.

Challenges:

1. Power estimate is a discrete-time event. Energy consumption is a continuous process with a higher correlation to battery drain.
It is possible to use Powertop to view the "power estimate" of a process/device/interrupt/timer.

Challenges:

1. Power estimate is a discrete-time event. Energy consumption is a continuous process with a higher correlation to battery drain.
2. Vendor-specific implementation
It is possible to use Powertop to view the "power estimate" of a process/device/interrupt/timer.

Challenges:

1. Power estimate is a **discrete-time event**. Energy consumption is a continuous process with a higher correlation to battery drain.
2. **Vendor-specific** implementation
3. **Actionability** of this data for end-users and programmers

**Process X consumes 1.45 Watts. What should the programmer do to optimize it?**
Outline

Background
Problem
Goal
Current Tools
PowerTOP
System Design
End Product
Conclusion
Device-Specific Measurements

Goal: Determine regression parameters
Device-Specific Measurements

Goal: Determine regression parameters

Algorithm:

1. Minimize system load by turning off all devices
2. Measure battery drain rate over multiple intervals
3. Turn on a single target device
4. Sweep target device parameters from low to high while measuring battery drain
5. Turn off target device or set parameter to low
6. Repeat step 3-5 for all target devices
Device-Specific Measurements

Goal: Determine regression parameters

Algorithm:
1. Minimize system load by turning off all devices
Device-Specific Measurements

Goal: Determine regression parameters

Algorithm:
1. Minimize system load by turning off all devices
2. Measure battery drain rate over multiple intervals
Goal: Determine regression parameters

Algorithm:
1. Minimize system load by turning off all devices
2. Measure battery drain rate over multiple intervals
3. Turn on a single target device
Device-Specific Measurements

Goal: Determine regression parameters
Algorithm:

1. Minimize system load by turning off all devices
2. Measure battery drain rate over multiple intervals
3. Turn on a single target device
4. Sweep target device parameters from low to high while measuring battery drain
Goal: Determine regression parameters
Algorithm:
1. Minimize system load by turning off all devices
2. Measure battery drain rate over multiple intervals
3. Turn on a single target device
4. Sweep target device parameters from low to high while measuring battery drain
5. Turn off target device or set parameter to low
Device-Specific Measurements

Goal: Determine regression parameters

Algorithm:
1. Minimize system load by turning off all devices
2. Measure battery drain rate over multiple intervals
3. Turn on a single target device
4. Sweep target device parameters from low to high while measuring battery drain
5. Turn off target device or set parameter to low
6. Repeat step 3-5 for all target devices
Kernel Process Accounting Infrastructure

Goal: Determine regression inputs
Goal: Determine regression inputs

Method:

Poll the process accounting infrastructure to determine CPU time allocation, network activity, open file handles, memory, disk usage, network, and screen wakeups.

Input the measured values in the regression model to predict energy consumption.
Goal: Determine regression inputs

Method:
- Poll the process accounting infrastructure to determine CPU time allocation, network activity, open file handles, memory, disk usage, network, and screen wakeups.
Goal: Determine regression inputs

Method:
- Poll the process accounting infrastructure to determine CPU time allocation, network activity, open file handles, memory, disk usage, network, and screen wakeups.
- Input the measured values in the regression model to predict energy consumption
System Design

- (One-time) Device-specific Measurements
- Kernel Process Accounting Infrastructure
- Multi-variate Regression Model
- Per-process Energy Consumption
Challenge: System Design

▶ Estimated value (All models are wrong, but some are useful.)
Estimated value (All models are wrong, but some are useful.)

Accuracy and Bias trade-off: Accurate models generate larger systemic load that biases observations
Challenge: Data Collection

There are millions of devices, and billions of ICs inside these devices. The power estimates can range across 2-3 orders of magnitude. How can we develop accurate & reliable power models across this diversity?
Challenge: Data Collection

There are millions of devices, and billions of ICs inside these devices. The power estimates can range across 2-3 orders of magnitude. How can we develop accurate & reliable power models across this diversity?

Privacy concern: Should users share this data to a "centralized" server?
Challenge: Validation of Correctness

- There is often significant difference between estimated values (from the model) and actual values (ground truth).
Challenge: Validation of Correctness

- There is often significant difference between estimated values (from the model) and actual values (ground truth).
- How to identify regressions from ground truth without hardware modifications?
Carbon emissions of software

Carbon Footprint = Energy Consumption × Energy Composition
Carbon emissions of software

Carbon Footprint = Energy Consumption $\times$ Energy Composition

Energy Consumption = Power $\times$ Latency
Carbon emissions of software

\[
\text{Carbon Footprint} = \text{Energy Consumption} \times \text{Energy Composition}
\]

\[
\text{Energy Consumption} = \text{Power} \times \text{Latency}
\]

\[
\text{Energy Composition} \text{ depends on multiple factors, including geography, time of availability, and cost}
\]
Outline

Background
Problem
Goal
Current Tools
  PowerTOP
System Design
End Product
Conclusion
End-users

Battery Level

Battery Usage

- Screen: 90%
- Wi-Fi: 33%
- Bluetooth: 6%
- System: 5%
- Inkscape: 4%
Expose API for programmers: Indicate devices with high energy consumption to allow backtracing to code

Example use-case: Energy-efficient code optimization suggestions in the coding platform
Expos API for system designers to enable better carbon accounting practices with clear scope identification.

**Example use-case:** Develop better tools to explore the design space of performance vs energy vs carbon efficiency.
Key Takeaways

▶ We cannot improve what we cannot measure.
Key Takeaways

▶ We cannot improve what we cannot measure.

▶ Non-CPU system components can dominate the overall energy consumption.
Thank you!

Feedback/Collaboration ?
https://www.linkedin.com/in/adityamanglik/
amangli@student.ethz.ch
Extended Discussion
How Does Energy Estimation Engine Work?

* the more accurate the energy attribution and estimation, the higher the engineering cost to collect hardware power characteristics
Reverse Engineering Windows’ Energy Estimation Engine: back-end

- The Energy Estimation Engine (E3) service runs on all Windows devices and attributes energy consumption to individual hardware components and applications.
The Energy Estimation Engine (E3) service runs on all Windows devices and attributes energy consumption to individual hardware components and applications.

Why software-based attribution: Few PCs in the market have such dedicated chips: According to reports, 99% of current devices in market lack dedicated current and voltage monitors.
Reverse Engineering Windows’ Energy Estimation Engine: back-end

▶ The Energy Estimation Engine (E3) service runs on all Windows devices and attributes energy consumption to individual hardware components and applications.

▶ Why software-based attribution: Few PCs in the market have such dedicated chips: According to reports, 99% of current devices in market lack dedicated current and voltage monitors.

▶ Software-based power attribution provides about 85% accuracy compared to a 98% accuracy rate from systems equipped with dedicated current and voltage monitors (e.g., Microsoft Surface)

▶ Microsoft claims that they prioritize data from devices with dedicated chips while developing the software-based power models.
Power profiles: Windows has separate power profiles for individual hardware devices like network, disks etc. Further, profiles specialize for Laptops, Tablets, Phones devices etc.

The following data columns can be observed in the E3 Service Report (shown below): ScreenOnEnergy, CPUEnergy, SoCEnergy, DisplayEnergy, DiskEnergy, MBBEnergy, NetworkEnergy, EmiEnergy, and many more.

**Figure: Data dump from E3 CLI**
The framework should be:

**Accurate**: Eliminate anomalous values

---

1Stretch goal
System design goals

The framework should be:

**Accurate**: Eliminate anomalous values

**Portable**: Able to function across different hardware vendors\(^2\)

\(^1\)Stretch goal
The framework should be:

**Accurate**: Eliminate anomalous values

**Portable**: Able to function across different hardware vendors\(^2\)

**Independent** of *extra* measurement devices

\(^1\)Stretch goal
System design goals

The framework should be:

**Accurate**: Eliminate anomalous values

**Portable**: Able to function across different hardware vendors

**Independent** of *extra* measurement devices

**Transparent**: Should not induce *any* load on the target system

---

1 Stretch goal
System design goals

The framework should be:

**Accurate**: Eliminate anomalous values

**Portable**: Able to function across different hardware vendors\(^2\)

**Independent** of *extra* measurement devices

**Transparent**: Should not induce *any* load on the target system\(^1\)

**Reliable**: Repeat experiments should yield *similar* results

---

\(^1\)Stretch goal

\(^2\)Stretch goal
Design Optimizations

Central information store to overcome randomness?

- Overcoming variation in values: Collect data across systems to create a database
Design Optimizations
Central information store to overcome randomness?

- Overcoming variation in values: Collect data across systems to create a database
- Privacy challenges: can we do better?
Design Considerations

- **Reliable:** Co-executing processes significantly influence power.

**Solution:** Energy consumption should be roughly similar.
Design Considerations

- Reliable: Co-executing processes significantly influence power. **Solution**: Energy consumption should be roughly similar.
- Accuracy:
  - Challenging to isolate individual contributions as many processes use multiple hardware devices simultaneously (CPU, GPU, Display, RAM, SSD, Ethernet/WiFi).
Design Considerations

- **Reliable:** Co-executing processes significantly influence power.
  
  **Solution:** Energy consumption should be roughly similar.

- **Accuracy:**
  
  - Challenging to isolate individual contributions as many processes use multiple hardware devices simultaneously (CPU, GPU, Display, RAM, SSD, Ethernet/WiFi).
  
  - Hardware devices do not measure/expose individual power draw.
Design Considerations

- Reliable: Co-executing processes significantly influence power.
  **Solution**: Energy consumption should be roughly similar.

- Accuracy:
  - Challenging to isolate individual contributions as many processes use multiple hardware devices simultaneously (CPU, GPU, Display, RAM, SSD, Ethernet/WiFi).
  - Hardware devices do not measure/expose individual power draw.
  - Significant variation in data-sheets across devices and vendors.
Design Considerations

- **Reliable:** Co-executing processes significantly influence power.  
  **Solution:** Energy consumption should be roughly similar.

- **Accuracy:**
  - Challenging to isolate individual contributions as many processes use multiple hardware devices simultaneously (CPU, GPU, Display, RAM, SSD, Ethernet/WiFi).
  - Hardware devices do not measure/expose individual power draw.
  - Significant variation in data-sheets across devices and vendors.
  - **Reliable values:** CPU perf counters (RAPL?) and current battery charge (ACPI?)
Different hardware devices

- CPU: Dominant factor, P-states vs C-states, interfaces (Intel RAPL)
- GPU: periodic bursts of large power draw
- RAM: Increasing DRAM capacity is challenging due to refresh power draw (Reference)
- I/O Peripherals: USB devices are polled every 5 ms
- Display: Often the most consistent drain
- Network Adaptors: Ethernet, WiFi ping frequency
- Disk: SSD, HDD writes are cached for bulk ops
Hardware requirements: Cannot rely on external power monitors

Transparency: Polling for values induces load on the target system

Able to function across different hardware vendor APIs

Actionability of data: Reporting hardware power values is "futile" because hardware is difficult to change, but processes might be optimized.