Modeling and Measuring Energy: from Hardware to Software

Adel Noureddine
Associate Professor HDR
University of Pau and Pays de l'Adour

Journées de Recherche en Apprentissage Frugal (JRAF 2024)
Grenoble, France

21 November 2024

UNIVERSITÉ
DE PAU ET DES
PAYS DE L'ADOUR

About me

Adel Noureddine

Associate Professor HDR in Computer Science

@University of Pau and Pays de l'Adour

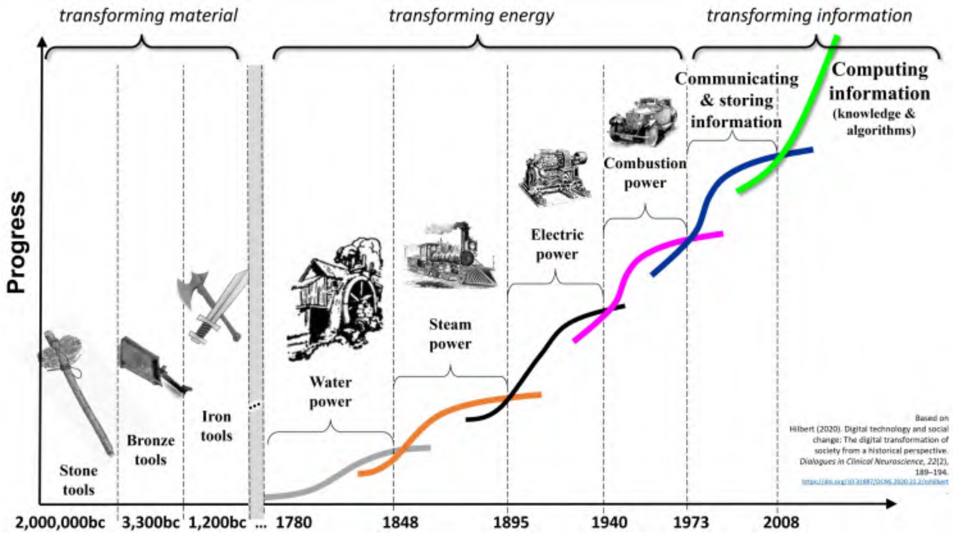
@LIUPPA laboratory

Researcher in Green IT, Software Engineering and Autonomic Computing

noureddine.org

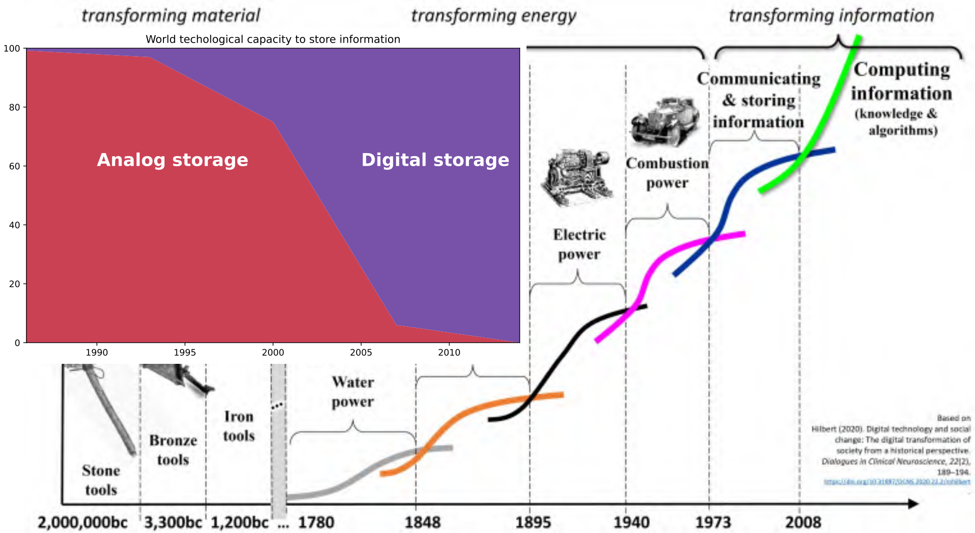
Context: Software Ecosystems

Kondratiev Waves of Innovation



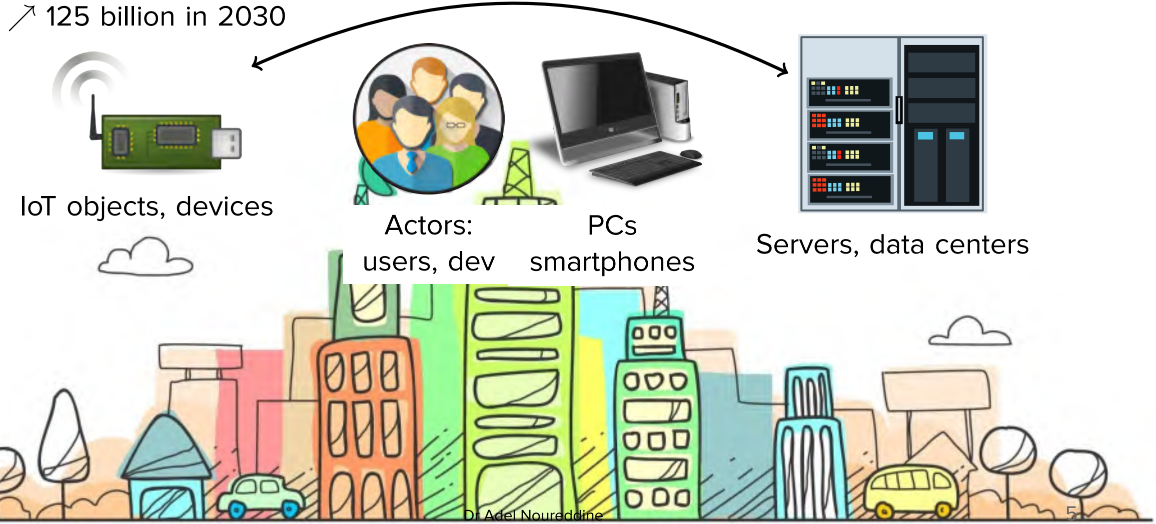
Based on
Hilbert (2020). Digital technology and social change: The digital transformation of society from a historical perspective. *Dialogues in Clinical Neuroscience, 22*(2), 189–194.
https://doi.org/10.31887/DCNS.2020.22.2/mhilbert

Kondratiev Waves of Innovation



Based on
Hilbert (2020). Digital technology and social change: The digital transformation of society from a historical perspective. *Dialogues in Clinical Neuroscience, 22*(2), 189–194.
https://doi.org/10.31887/DCNS.2020.22.2/mhilbert

Software Ecosystems



↗ 125 billion in 2030

IoT objects, devices

Actors:
users, dev

PCs
smartphones

Servers, data centers

Software Ecosystems

↗ 125 billion in 2030

IoT objects, devices

Actors:
users, dev

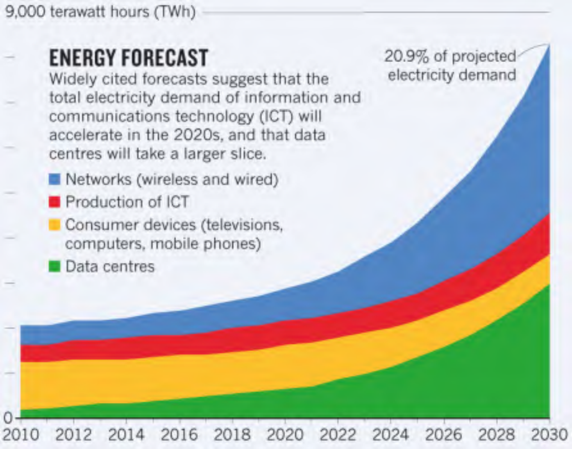
PCs
smartphones

Servers, data centers

Software

Our lives depend on software, our industry depends on software, our whole society depends on software (Software Heritage)

Environmental Impact of ICT



ENERGY FORECAST

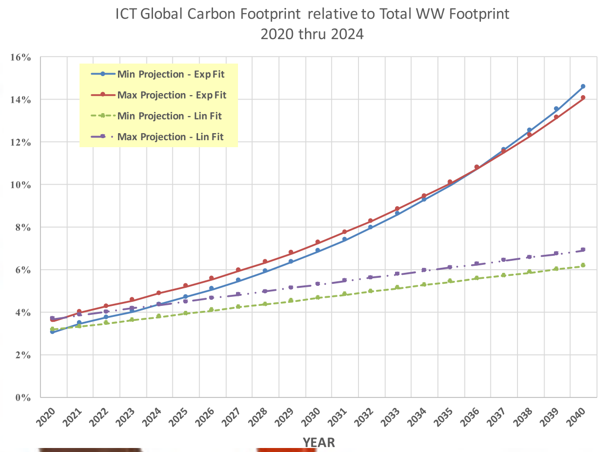
Widely cited forecasts suggest that the total electricity demand of information and communications technology (ICT) will accelerate in the 2020s, and that data centres will take a larger slice.

Electricity

7% in 2020

13% in 2030

Environmental Impact of ICT



GHGE

2% in 2007

14% in 2040

Complexity of Heterogeneity in Software Ecosystems

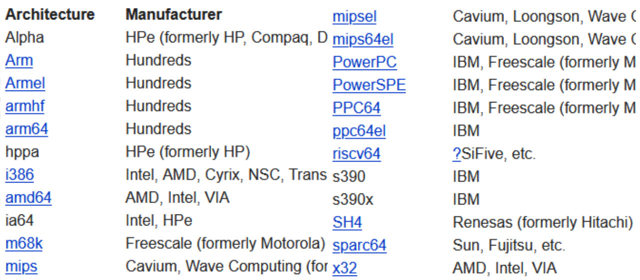
Hardware

Software

Humans

Complexity of Heterogeneity in Software Ecosystems

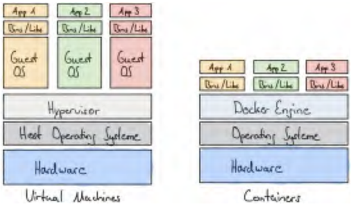
- **Hardware**: IoT to data centers, CPS

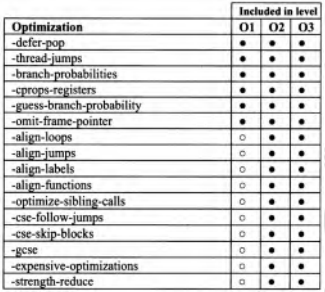


Architecture	Manufacturer
Alpha	HPe (formerly HP, Compaq, D
Arm	Hundreds
Armel	Hundreds
armhf	Hundreds
arm64	Hundreds
hppa	HPe (formerly HP)
i386	Intel, AMD, Cyrix, NSC, Trans
amd64	AMD, Intel, VIA
ia64	Intel, HPe
m68k	Freescale (formerly Motorola)
mips	Cavium, Wave Computing (for
mipsel	Cavium, Loongson, Wave (
mips64el	Cavium, Loongson, Wave (
PowerPC	IBM, Freescale (formerly M
PowerSPE	IBM, Freescale (formerly M
PPC64	IBM, Freescale (formerly M
ppc64el	IBM
riscv64	?SiFive, etc.
s390	IBM
s390x	IBM
SH4	Renesas (formerly Hitachi)
sparc64	Sun, Fujitsu, etc.
x32	AMD, Intel, VIA

Complexity of Heterogeneity in Software Ecosystems

- **Hardware**: IoT to data centers, CPS
- **Software**: OS, VM, software versions, updates, configurations

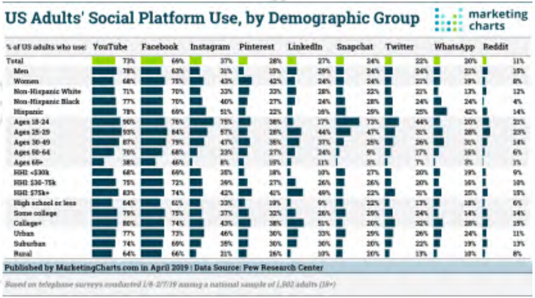




Optimization	Included in level		
	O1	O2	O3
-defer-pop	●	●	●
-thread-jumps	●	●	●
-branch-probabilities	●	●	●
-cprops-registers	●	●	●
-guess-branch-probability	●	●	●
-omit-frame-pointer	●	●	●
-align-loops	○	●	●
-align-jumps	○	●	●
-align-labels	○	●	●
-align-functions	○	●	●
-optimize-sibling-calls	○	●	●
-cse-follow-jumps	○	●	●
-cse-skip-blocks	○	●	●
-gcse	○	●	●
-expensive-optimizations	○	●	●
-strength-reduce	○	●	●

Complexity of Heterogeneity in Software Ecosystems

- **Hardware**: IoT to data centers, CPS
- **Software**: OS, VM, software versions, updates, configurations
- **Humans**: user profiles, developers, system admins, deciders, procurers, managers, etc.



US Adults' Social Platform Use, by Demographic Group

% of US adults who use:	YouTube	Facebook	Instagram	Pinterest	LinkedIn	Snapchat	Twitter	WhatsApp	Reddit
Total	73%	69%	37%	28%	27%	24%	22%	20%	11%
Men	78%	63%	31%	15%	29%	24%	24%	21%	15%
Women	68%	75%	43%	42%	24%	24%	21%	19%	8%
Non-Hispanic White	71%	70%	33%	33%	28%	22%	21%	13%	12%
Non-Hispanic Black	77%	70%	40%	27%	24%	28%	24%	24%	4%
Hispanic	78%	69%	51%	22%	16%	29%	25%	42%	14%
Ages 18-24	90%	76%	75%	38%	17%	73%	44%	20%	21%
Ages 25-29	93%	84%	57%	28%	44%	47%	31%	28%	23%
Ages 30-49	87%	79%	47%	35%	37%	25%	26%	31%	14%
Ages 50-64	70%	68%	23%	27%	24%	9%	17%	16%	6%
Ages 65+	38%	46%	8%	15%	11%	3%	7%	3%	1%
HHI: <$30k	68%	69%	35%	18%	10%	27%	20%	19%	9%
HHI: $30-75k	75%	72%	39%	27%	26%	26%	20%	16%	10%
HHI: $75k+	83%	74%	42%	41%	49%	22%	31%	25%	15%
High school or less	64%	61%	33%	19%	9%	22%	13%	18%	6%
Some college	79%	75%	37%	32%	26%	29%	24%	14%	14%
College+	80%	74%	43%	38%	51%	20%	32%	28%	15%
Urban	77%	73%	46%	30%	33%	29%	26%	24%	11%
Suburban	74%	69%	35%	30%	30%	20%	22%	19%	13%
Rural	64%	66%	21%	26%	10%	20%	13%	10%	8%

Published by MarketingCharts.com in April 2019 | Data Source: Pew Research Center

Based on telephone surveys conducted 1/8-2/7/19 among a national sample of 1,502 adults (18+)

- Project manager, scrum master
- Developers, architect, designer
- Tester, analyst, QA

Scientific Challenges

Observing Energy

- Monitoring energy in heterogeneous environments?
- Mapping energy across software & hardware layers?
- Providing energy monitoring data to users?

Understanding Impacts on Energy

- Understanding hardware impacts?
- Understanding software & source code impacts?
- Understanding the role of users?

Holistic approach:
Observe & understand software energy ubiquitously (*everywhere*)

Scientific Challenges

Observing Energy

- Monitoring energy in heterogeneous environments?
- Mapping energy across software & hardware layers?
- Providing energy monitoring data to users?

Understanding Impacts on Energy

- Understanding hardware impacts?
- Understanding software & source code impacts?
- Understanding the role of users?

Holistic approach:
Observe & understand software energy ubiquitously (*everywhere*)

Scientific Challenges

Observing Energy

- Monitoring energy in heterogeneous environments?
- Mapping energy a hardware layers?
- Providing energy users?

Understanding Impacts on Energy

- Understanding hardware impacts?
- Understanding software & source
- the role of users?

Limitations of existing approaches:

Use case specific

Not covering ecosystem

Not integrating human actors

Holistic approach:
Observe & understand software energy ubiquitously (*everywhere*)

Scientific Challenges

Observing Energy

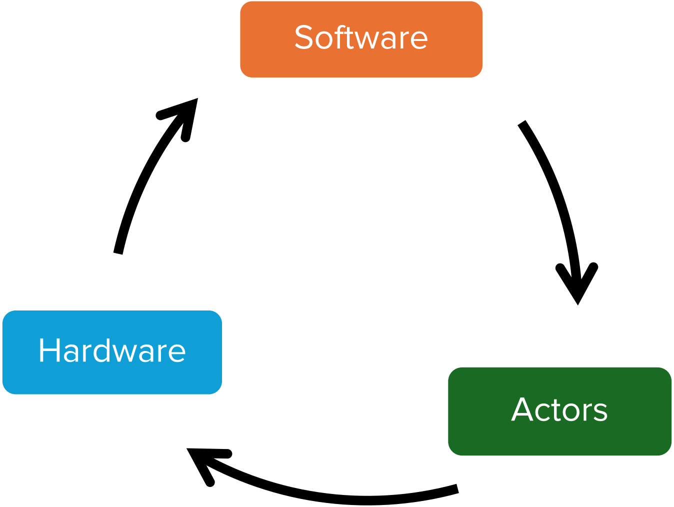
- Monitoring energy in heterogeneous environments?
- Mapping energy across software & hardware layers?
- Providing energy monitoring data to users?

Understanding Impacts on Energy

- Understanding hardware impacts?
- Understanding software & source code impacts?
- Understanding the role of users?

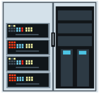
Holistic approach:
Observe & understand software energy ubiquitously (*everywhere*)

Observing Green Software Ecosystems



Observing Green Software Ecosystems

Server



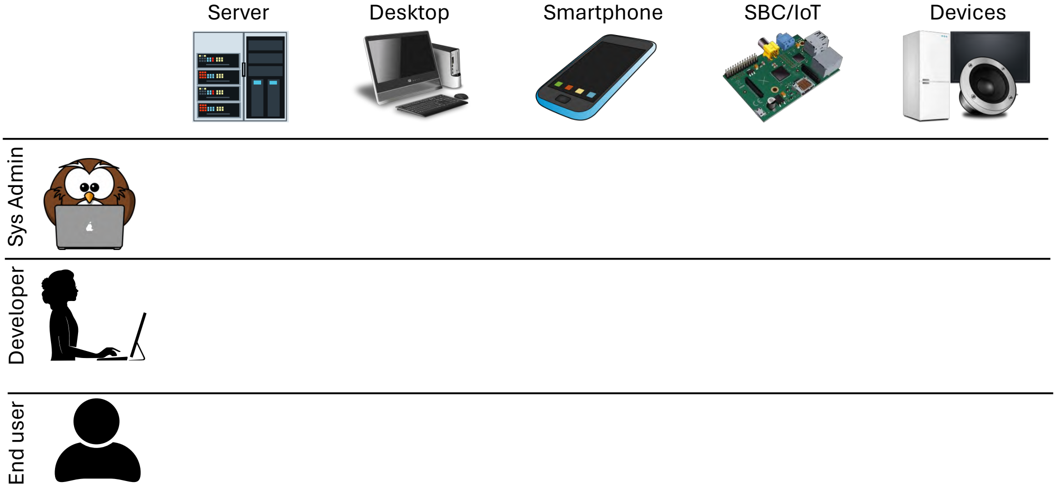
Desktop

Smartphone

SBC/IoT

Devices

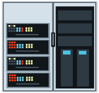
Observing Green Software Ecosystems



	Server	Desktop	Smartphone	SBC/IoT	Devices
Sys Admin					
Developer					
End user					

Observing Green Software Ecosystems

Server



Desktop

Smartphone

SBC/IoT

Devices

Sys Admin

Developer

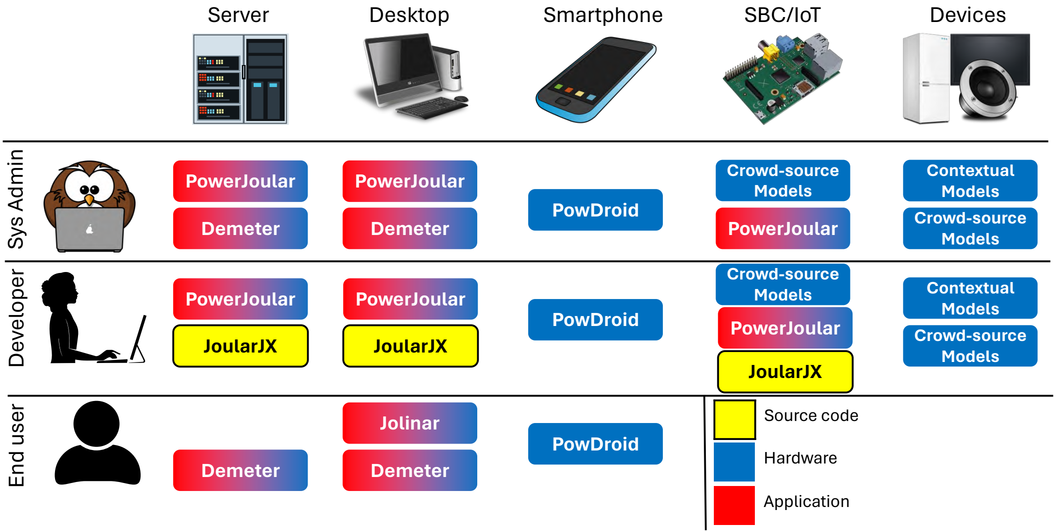
End user

🟨 Source code

🟦 Hardware

🟥 Application

Observing Green Software Ecosystems



	Server	Desktop	Smartphone	SBC/IoT	Devices
Sys Admin	PowerJoular Demeter	PowerJoular Demeter	PowDroid	Crowd-source Models PowerJoular	Contextual Models Crowd-source Models
Developer	PowerJoular JoularJX	PowerJoular JoularJX	PowDroid	Crowd-source Models PowerJoular JoularJX	Contextual Models Crowd-source Models
End user	Demeter	Jolinar Demeter	PowDroid		

🟨 Source code

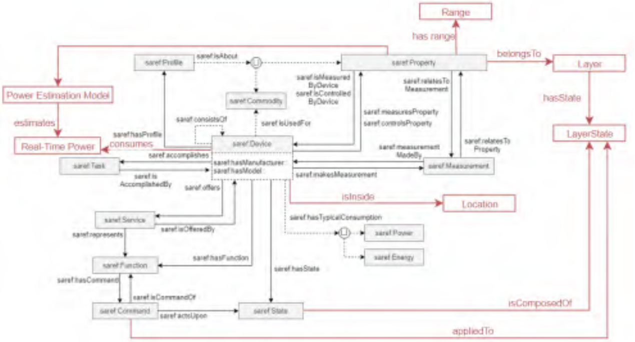
🟦 Hardware

🟥 Application

Modeling power consumption of devices

Contextual Modeling

- Devices, software & humans
- Actors & connectivity generate contextual data
- Map and observe the energy dimension
- Green extension of the SAREF ontology

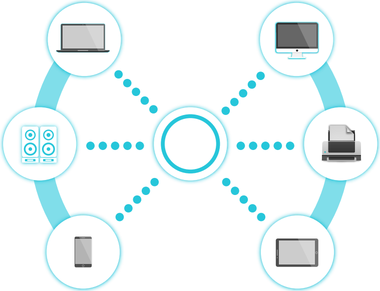


PhD thesis: Houssam Kanso, 2019-2022

An Automated Energy Management Framework for Smart Homes. Kanso et al. In JAISE journal. 2023

Billions of heterogeneous devices

- 100+ billion connected devices in 2030
- Heterogeneity in hardware architecture, configuration & software
- Real-time monitoring without hardware power meters
- Current power models are not evolutive, or are built on static data sets



Modeling approach

- Expert identifies a characteristic to model
  - CPU utilization, size of data written to disk, time spend sending data in network, etc.
- Build or use a benchmark to stress the characteristic
  - Example: for CPU utilization, stress every percentage of the CPU (from 0% to 100%)
- Collect power measurements & apply statistical approaches
- Generate power model (often regression models)
- Validate power models (cross-validation, margin of error)

Modeling approach

- Expert identifies a characteristic to model
  - CPU utilization, size of data written to disk, time spend sending data in network, etc.
- Build or use a benchmark to stress the characteristic
  - Example: for CPU utilization, stress every percentage of the CPU (from 0% to 100%)
- Collect power measurements & apply statistical approaches
- Generate power model (often regression models)
- Validate power models (cross-validation, margin of error)

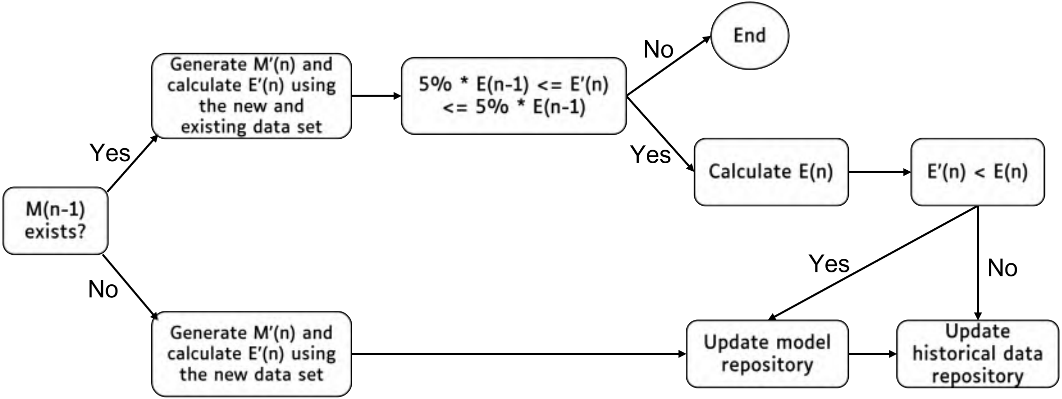
Approach

Automate and crowd-source

Implementation for Raspberry Pi devices

- Characteristic to model: CPU utilization
- Collected metrics: CPU utilization (calculated from CPU cycles), and real power consumption (from a powermeter)
- Cleaning and synchronization of collected data (irrelevant data points, synchronize clock diversion with power meter and raspberry, timestamp, etc.)

Model Generator & Validator

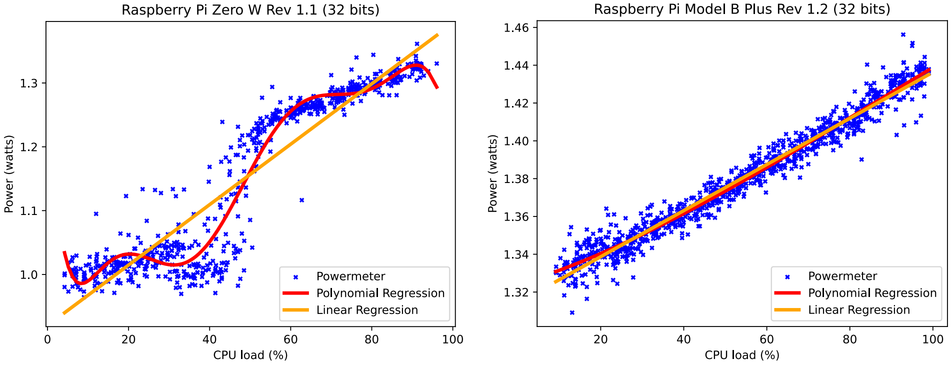

M(n-1) exists?
Yes
Generate M'(n) and calculate E'(n) using the new and existing data set
5% * E(n-1) <= E'(n) <= 5% * E(n-1)
No
End
Yes
Calculate E(n)
E'(n) < E(n)
Yes
No
No
Generate M'(n) and calculate E'(n) using the new data set
Update model repository
Update historical data repository

Empirical validation

- 8 Raspberry Pi devices from all generations, 32 and 64 bits (since this study, we added RPi 400, RPi 5 & Asus TinkerBoard S)
- PowerSPY2 powermeter, disconnect all external peripherals including USB and HDMI ports

Model	Rev.	OS	CPU Architecture	Cores	Released
Zero W	1.1	32	armv6l	1	2017
1B	2	32	armv6l	1	2012
1B+	1.2	32	armv6l	1	2014
2B	1.1	32	armv7l	4	2015
3B	1.2	32	armv7l	4	2016
3B+	1.3	32	armv7l	4	2018
4B	1.1	32/64	armv7l/aarch64	4	2019
4B	1.2	32/64	armv7l/aarch64	4	2019

Power models

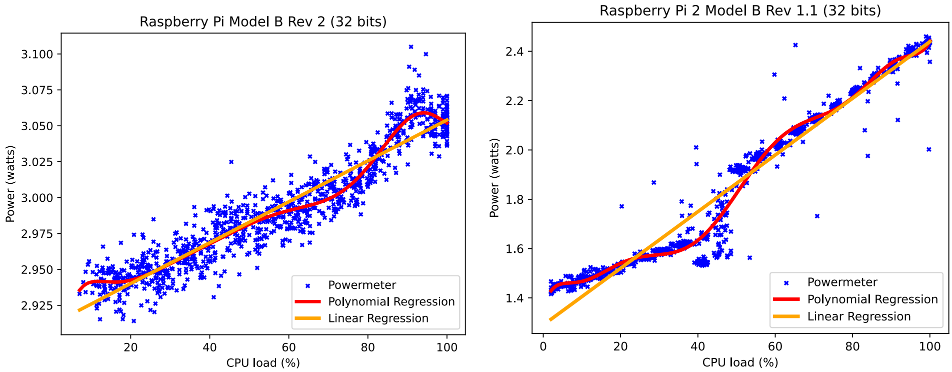


(a)

(b)

(a) Raspberry Pi Zero W Rev 1.1 (32 bits) (b) Raspberry Pi Model B Plus Rev 1.2 (32 bits)

Power models

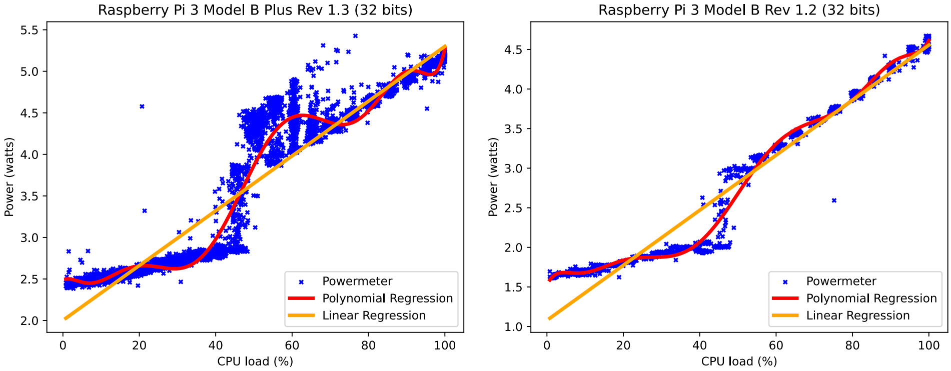


(a)

(b)

(a) Raspberry Pi Model B Rev 2 (32 bits) (b) Raspberry Pi 2 Model B Rev 1.1 (32 bits)

Power models

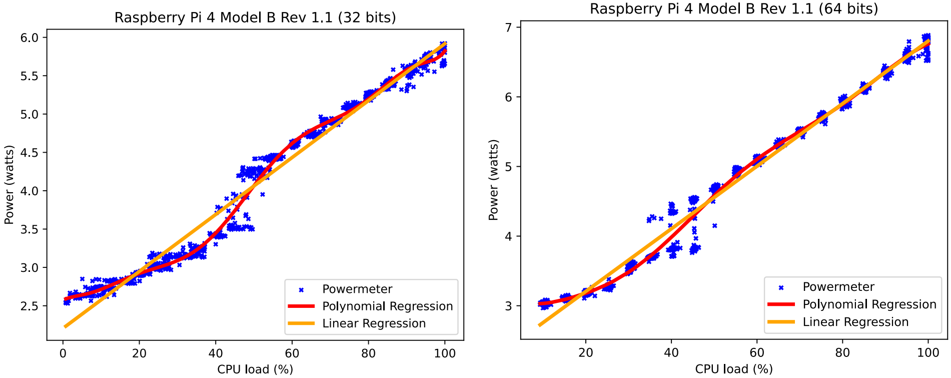


(a)

(b)

(a) Raspberry Pi 3 Mobel B Plus Rev 1.3 (32 bits) (b) Raspberry Pi 3 Model B Rev 1.2 (32 bits)

Power models


Raspberry Pi 4 Model B Rev 1.1 (32 bits)

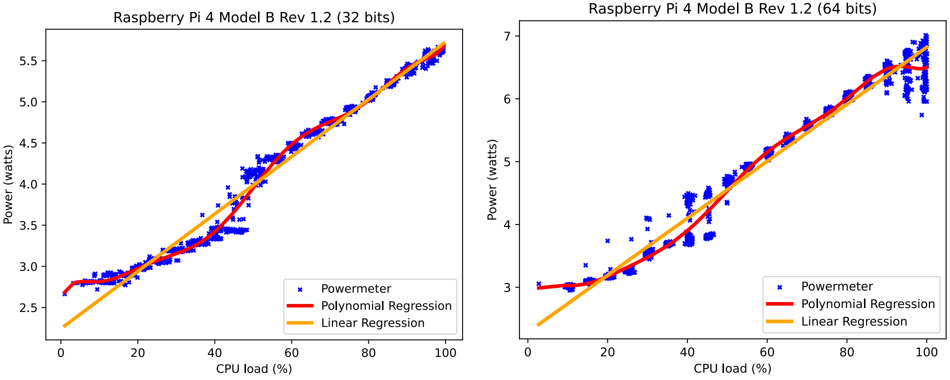
Raspberry Pi 4 Model B Rev 1.1 (64 bits)

(a)

(b)

(a) Raspberry Pi 4 Model B Rev 1.1 (32 bits) (b) Raspberry Pi 4 Model B Rev 1.1 (64 bits)

Power models

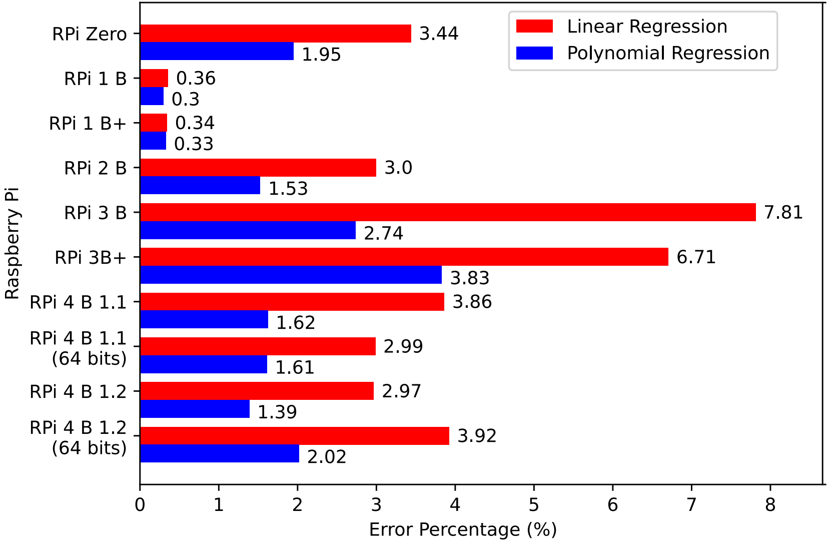


(a)

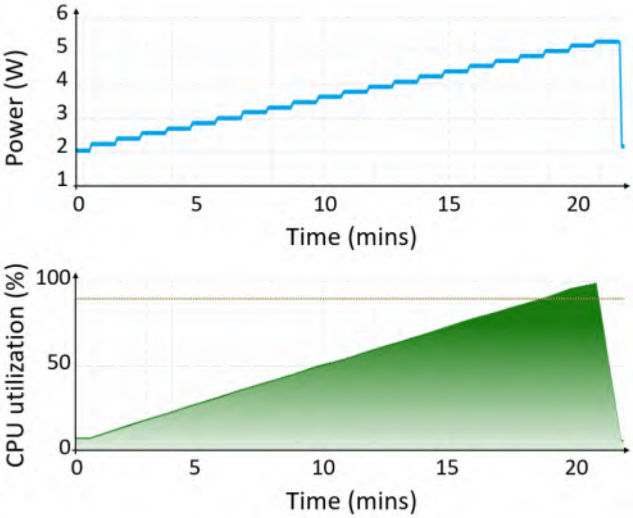
(b)

(a) Raspberry Pi 4 Model B Rev 1.2 (32 bits) (b) Raspberry Pi 4 Model B Rev 1.1 (64 bits)

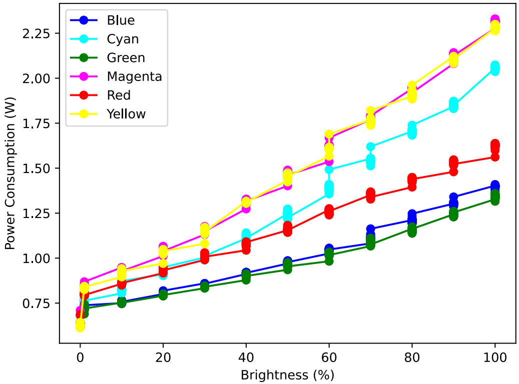
Power models accuracy



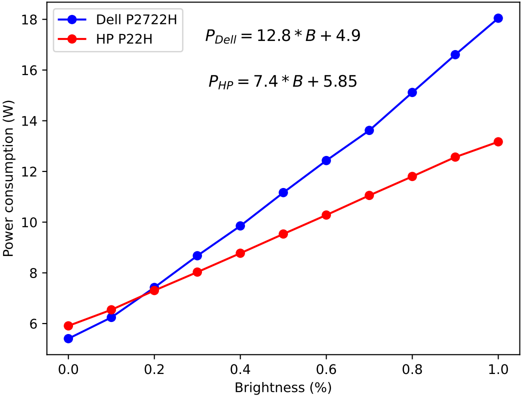
Usage: remote power monitoring with Zabbix server



Beyond Computing Devices



Light bulb



Monitors

Power Models Database

- A database containing power models for various computing components, and hardware devices
- Currently, multiple models for single board computers (Raspberry Pi, Asus Tinker Board), and monitors
- Aims to centralize power models with an open data format
- Also aims to provide an API to download or update power models
- github.com/joular/powermodels

Power Models Database: Raspberry Pi devices

Model	Codename	Component	Model Variable
Model Zero W (rev 1.1), 32-bit OS	rbpzw1.1	CPU	CPU usage
Model 1 B (rev 2), 32-bit OS	rbp1b2	CPU	CPU usage
Model 1 B+ (rev 1.2), 32-bit OS	rbp1b+1.2	CPU	CPU usage
Model 2 B (rev 1.1), 32-bit OS	rbp2b1.1	CPU	CPU usage
Model 3 B (rev 1.2), 32-bit OS	rbp3b1.2	CPU	CPU usage
Model 3 B+ (rev 1.3), 32-bit OS	rbp3b+1.3	CPU	CPU usage
Model 4 B (rev 1.1), 32-bit OS	rbp4b1.1	CPU	CPU usage
Model 4 B (rev 1.1), 64-bit OS	rbp4b1.1-64	CPU	CPU usage
Model 4 B (rev 1.2), 32-bit OS	rbp4b1.2	CPU	CPU usage
Model 4 B (rev 1.2), 64-bit OS	rbp4b1.2-64	CPU	CPU usage
Model 400 (rev 1.0), 64-bit OS	rbp4001.0-64	CPU	CPU usage
Model 5 B (rev 1.0), 64-bit OS	rbp5b1.0-64	CPU	CPU usage

Power Models Database: other devices & hardware

Model	Codename	Component	Model Variable
Asus Tinker Board (S)	asustbs	CPU	CPU usage
Dell P2722H	p2722h	Entire device	Brightness
HP P22H	p22h	Entire device	Brightness

Software energy

What is software energy?

- Software energy is the energy consumed by hardware components to execute software instructions
- Example : a software instructs the processor to calculate the first 100 digits of Pi
- Software energy = energy consumed by hardware components (CPU, memory, etc.) for the calculations

Software energy

Accurately measuring software energy is tricky and difficult:

- Source code energy is very hard to predict. Ex. What is the energy cost of Towers of Hanoi algorithm?
- Energy is measured on runtime, and depends on hardware configuration. Same software → different energy consumption depending on hardware (mobile vs. server)
- Energy is affected by more than just hardware configuration : temperature, materials, other software running, etc.
- As hardware is the one consuming energy, there is no physical devices or meters to measure software energy directly

How to measure software energy?

Software energy can be estimated with a software approach :

- Model hardware’s power with a power model (RAPL, regression models, etc.)
- Calculates software’s hardware usage from available metrics (CPU usage from the OS, network usage with a profiler, etc.)
- Allocate software energy according to hardware usage

PowerJoular

PowerJoular

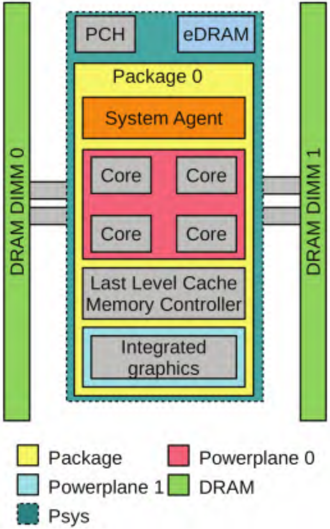
- PowerJoular is a command line software to monitor, in real time, the power consumption of software and hardware components (CPU, Nvidia GPU)
- Support multiple CPU architectures:
  - ▶ x86/64 using Intel RAPL interface (Intel, AMD) through `powercap`
  - ▶ ARM (most Raspberry Pi, including RPi 5, and Asus Tinker Board) through our power model
  - ▶ Inside virtual machines in all supported host platforms
- Low overhead (Ada, compiled to native code), GPL 3
- noureddine.org/research/joular/powerjoular ↗
- github.com/joular/powerjoular ↗

PowerJoular: Intel/AMD CPU

- Uses Intel RAPL interface through powercap (integrated in the Linux kernel)
- Reading `energy_uj` files in `/sys/class/powercap/intel-rapl/` folder
- `energy_uj` provides an increased energy value in µJ (difference of `energy_uj` between two timestamps →energy consumed by the CPU)

Steps:

- Detects which RAPL domain is supported by the CPU
- if `psys` is supported, we use it
- if not, we use `pkg`



PowerJoular: ARM

- Supports a list of Raspberry and Asus Tinker Board devices
- Uses our power models based on CPU utilization (collected from `/proc/stat`)
- By default: polynomial model (up to degree 9), but linear can be used

Example for RPi 4 (where x is CPU utilization):

$$P(x) = 2.58542069543335 + 12.335449x - 248.010554x^2 + 2379.832320x^3 - 11962.419149x^4 + 34444.268647x^5 - 58455.266502x^6 + 57698.685016x^7 - 30618.557703x^8 + 6752.265368x^9$$

PowerJoular: virtual machines

- All its functionalities (such as monitoring a PID or an application) work the same inside a virtual machine as with bare metal installation
- PowerJoular in the guest OS needs to get the power consumption of the virtual machine instance itself
- The power data of the VM process need to be written to a shared file between the host and the guest
- PowerJoular is agnostic to what power tools in installed in the host and can work with any available tool that is capable of monitoring the VM process

Documention: joular.github.io/powerjoular/ref/vm.html ↗

PowerJoular: Nvidia GPU

- Support monitoring GPU energy by using Nvidia SMI interface

```
nvidia-smi --format=csv,noheader,nounits --query-gpu=power.
    draw
```

PowerJoular: Process Monitoring

- PowerJoular can monitor the energy of a specific process using its Process ID (`PID`)
- Collects CPU utilization and `PID` utilization from `/proc/stat` and `/proc/pid/stat`
- Allocate `PID` energy, every second in real time, according to its CPU cycles usage percentage

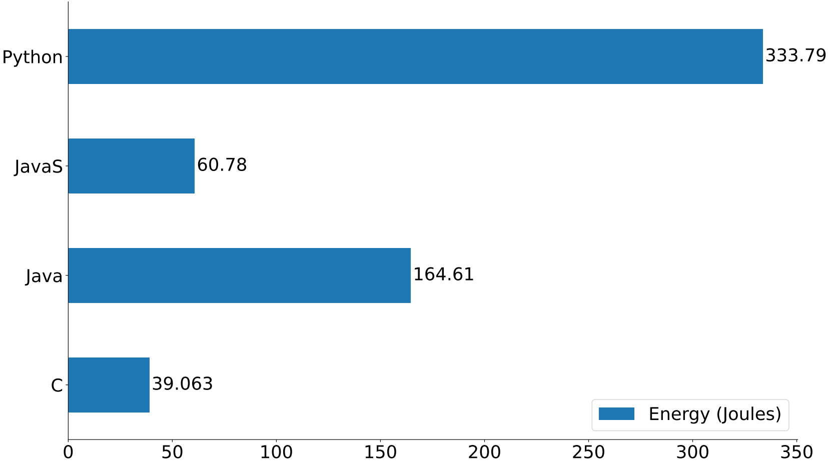
PowerJoular: Application Monitoring

- PowerJoular can monitor the energy of a specific application
- Uses the application name to seach for all its PIDs (through `pidof` command in Linux, or `pgrep`)
- Measures and aggregate the energy of all PIDs in real time (*sum of the energy of all app's PIDs*)
- PowerJoular can keep up with process creation and destruction by applications (checks for application's PIDs every second)

PowerJoular: additional features

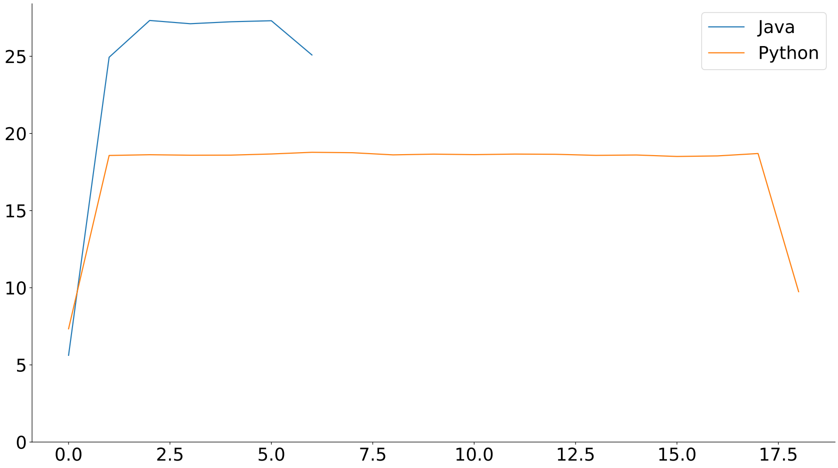
- Provides a systemd service to automate energy monitoring
- Can provide energy for non elevated users without weakening the system (such as with direct RAPL access, *i.e.*, PLATYPUS attack)
- Writes energy results to CSV files, for every second
- Has an overwrite mode: only last measurement is stored in file

PowerJoular: example of usage



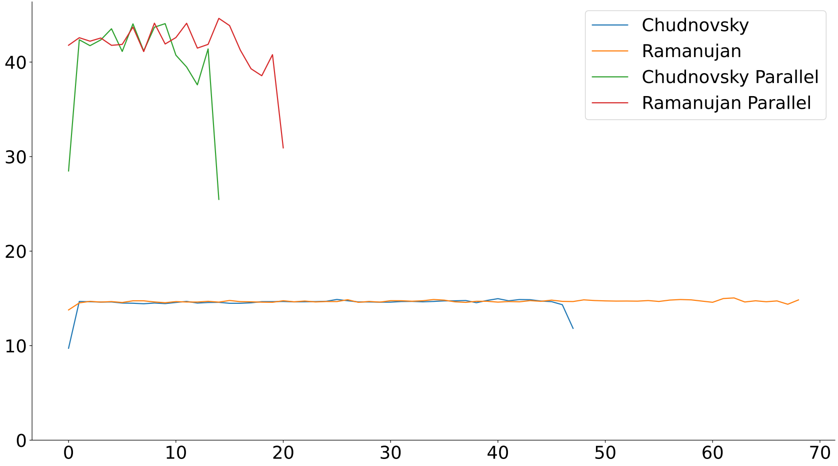
Example of energy consumption of the Ray casting algorithm

PowerJoular: example of usage



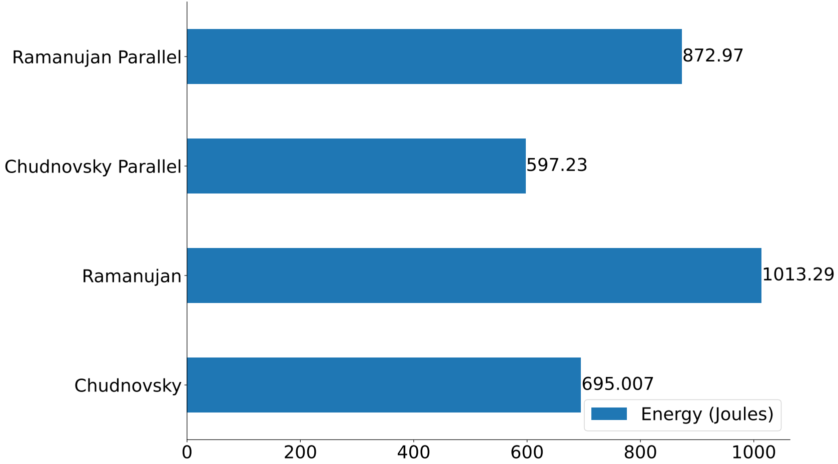
Example of energy consumption of the Ray casting algorithm

PowerJoular: example of usage



Example of energy consumption of y-cruncher to calculate 150m digits of Pi

PowerJoular: example of usage



Example of energy consumption of y-cruncher to calculate 150m digits of Pi

Source code energy with JoularJX

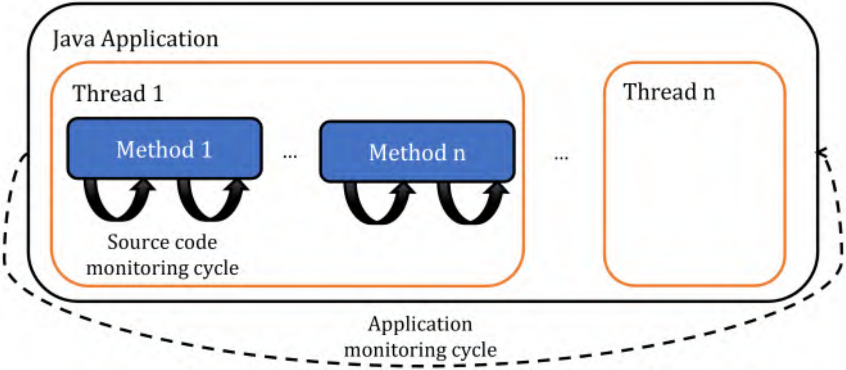
Source code energy

- How to detect and monitor energy consumption ”inside” applications?
- What is the energy cost of individual methods, classes, etc.?

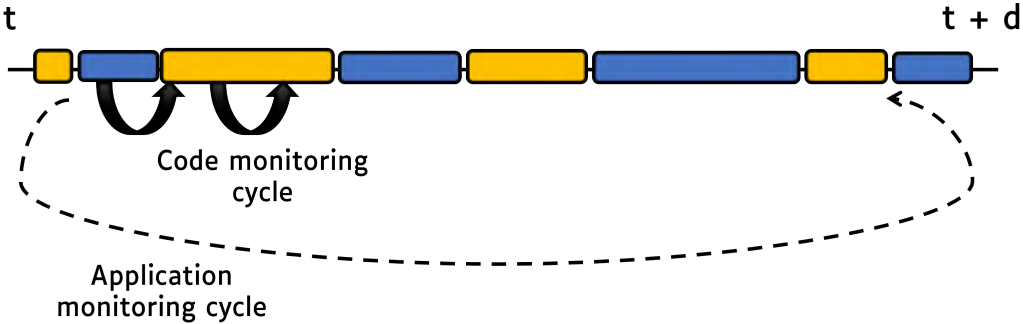
Statistical sampling

- Monitor and collect metrics from the operating system, virtual machine, software, traces, etc.
- Apply statistical methods to estimate energy consumption of source code
- Have a lower overhead than instrumentation or annotations
- In some cases, doesn't require the source code (*e.g.*, in Java)
- Example tool: JoularJX

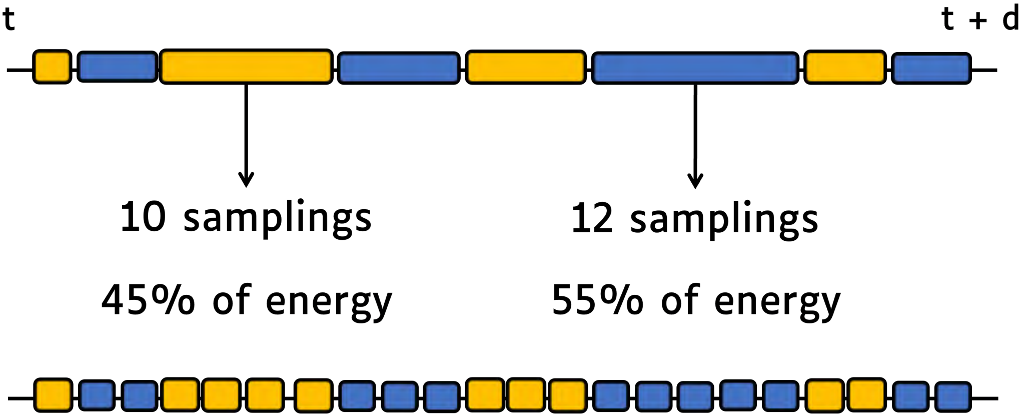
Statistical sampling: JoularJX approach



Statistical sampling: JoularJX approach



Statistical sampling: JoularJX approach



t

t + d

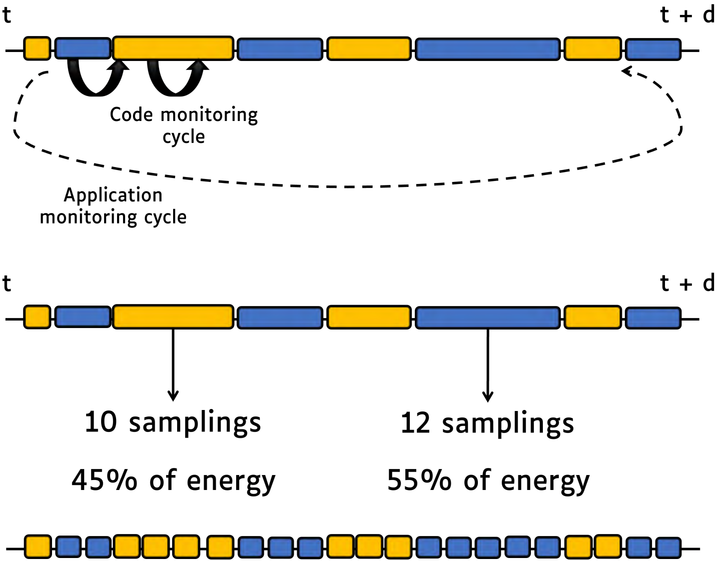
10 samplings

45% of energy

12 samplings

55% of energy

Statistical sampling: JoularJX approach



JoularJX

- JoularJX is a Java agent for software power monitoring at the source code level
- Support multiple architectures: x86/64 (Intel/AMD), Intel & ARM on macOS, ARM on Raspberry Pi and Asus Tinker Board, and in virtual machines
- GPL3 and works on Windows, macOS and Linux
- Real time power monitoring of the source code (methods and execution branches)
- noureddine.org/research/joular/joularjx
- github.com/joular/joularjx

JoularJX

- Measures the energy for every method of the application and/or the JDK
- Measures the energy for methods' call tree (all execution branches)
- Monitor the power consumption evolution of every method
- Monitors in real time and exposes all monitored data in CSV files
- No modifications needed nor access to the application's source code

JoularJX: CPU power

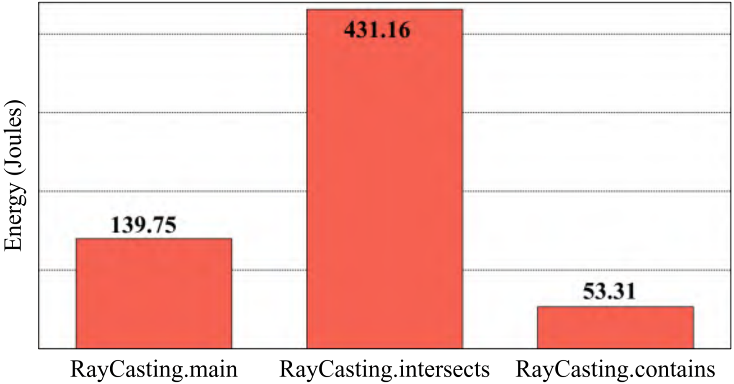
- Get CPU energy from RAPL on x86/64 Linux and Windows
- In x86/64 Linux: uses `powercap` and a similar approach to PowerJoular
- On x86/64 Windows:
  - Older versions of JoularJX: uses Intel Power Gadget API, only for Intel CPUs (deprecated by Intel)
  - Latest version: uses Hubblo's RAPL driver for Windows (reads MSR's like `powercap`, works for Intel/AMD)
- On ARM Linux: our own power models for Raspberry Pi and Tinker Board (similar to PowerJoular)
- on macOS (Intel & Apple's ARM chips): calls and parses `powermetrics` command

JoularJX: virtual machines

- JoularJX also works inside virtual machines. All its functionalities work the same inside a virtual machine as with bare metal installation
- In virtual machines, JoularJX in the guest OS needs to get the power consumption of the virtual machine instance itself
- JoularJX is agnostic to what power tools in installed in the host and can work with any available tool that is capable of monitoring the VM process

Documention: joular.github.io/joularjx/ref/vm.html ↗

JoularJX: example of usage

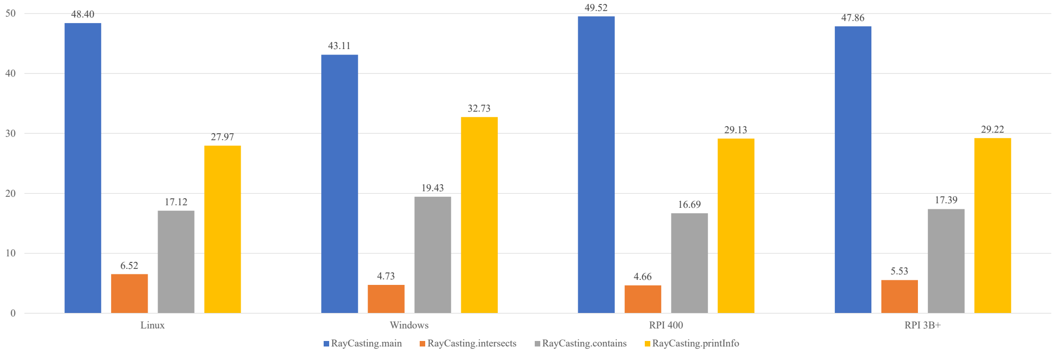


Example of energy consumption of the Java implementation of the Ray casting methods

JoularJX

Energy of methods

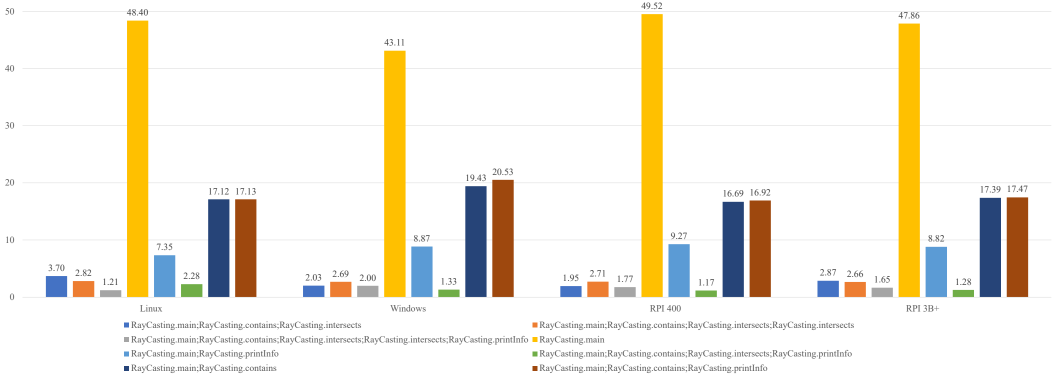
Energy Consumption of Methods in RayCasting Java Program, in percentage



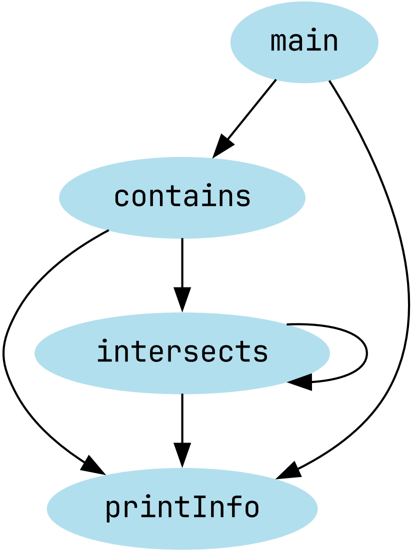
JoularJX

Energy of execution branches

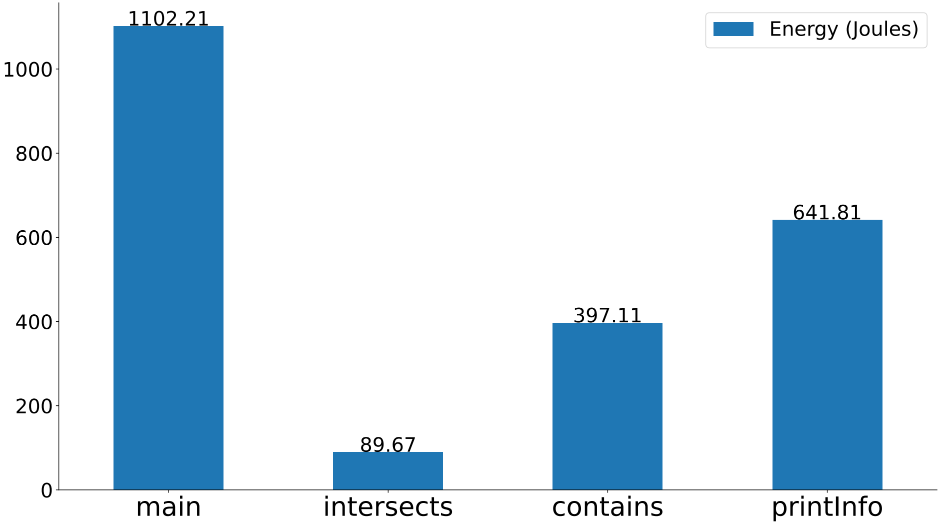
Energy Consumption of Methods' Call Tree in RayCasting Java Program, in percentage



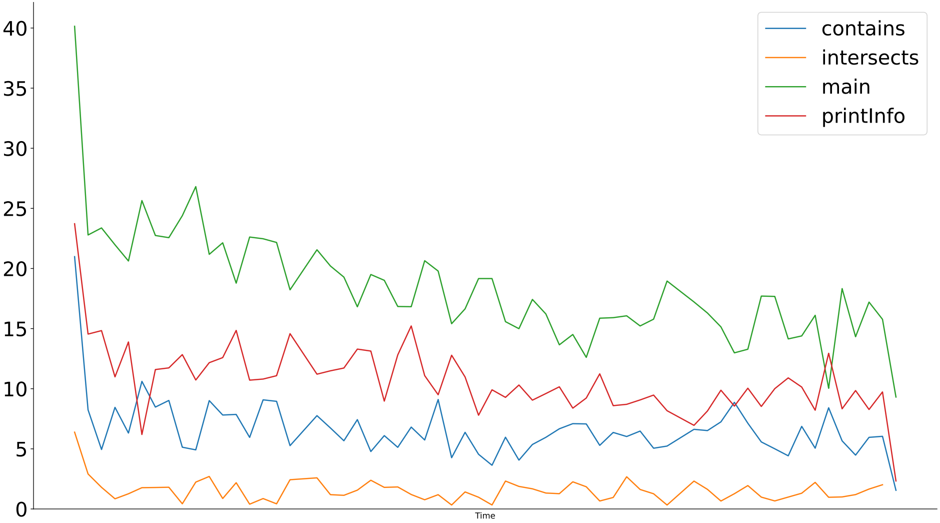
Software example: custom RayCasting algorithm



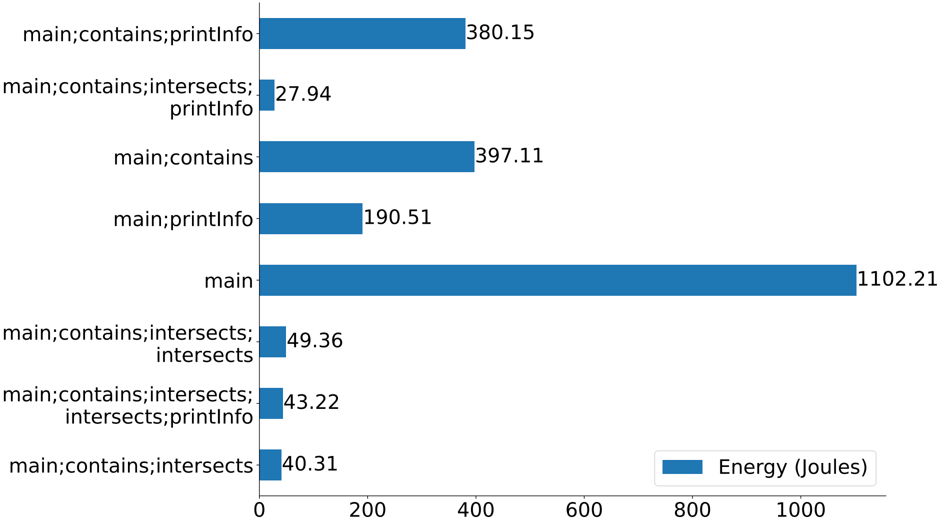
Energy consumption of methods



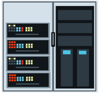
Power evolution of methods



Energy conusmption of the call tree



Summary

		Server 	Desktop 	Smartphone 	SBC/IoT 	Devices
Sys Admin		PowerJoular Demeter	PowerJoular Demeter	PowDroid	Crowd-source Models PowerJoular	Contextual Models Crowd-source Models
Developer		PowerJoular JoularJX	PowerJoular JoularJX	PowDroid	Crowd-source Models PowerJoular JoularJX	Contextual Models Crowd-source Models
End user		Demeter	Jolinar Demeter	PowDroid		

🟨 Source code

🟦 Hardware

🟥 Application

Analyzing Software Energy Consumption. Noureddine. Technical Briefing @ICSE 2024 conference

About me

Adel Noureddine

Associate Professor HDR in Computer Science

@University of Pau and Pays de l’Adour

@LIUPPA laboratory

Researcher in Green IT, Software Engineering and Autonomic Computing

noureddine.org