ENeRgy sustaInability COding, a practical use case

Benoit Lange
Benoit.lange@inria.fr

24 March 2025

OUTLINE

1. Introduction
2. Motivations
3. The platform
4. Results
5. Discussions

Introduction

The impact of computer engineering on climate is now important

The number of application is growing quickly

➔ More data center

➔ More energy

In a close future power plant will be dedicated to data centers..

But Green computing can be now used

-> but mainly focused on hardware and data center

Motivations

Hardware

- Computation capabilities is increasing
  Energy consumption decreasing
- New hardware functionalities
  SSE, AVX, AVX 512, ..
- New kind of architecture
- “New” kind of computation hardware :
  GPU, FPGA, TPU, LPU
- New IO hardware

Data centers

- 1% of total energy consumption
- Between 2000 and 2005, energy consumption double
- Increased of 6% between 2010 and 2018

Embedded or Mobile Software development

- Optimize the Energy of IoT devices
- Optimize communication to reduce the energy consumptions
- Optimize mobile software

Motivations

“Classic” Software development

- BUT …
  “I don’t care”
  “I don’t know how to do that”
  “It is too complex; we have already software to develop …”
  “Servers have more and more resources, we can use them”
  “My customer is highly constrained by time”
  “The service must run 24/7”

- Existing solutions are too complex
  ➔ ENRICO platform

➔ RQ : Is it possible to drive the evolution of an application with energy metrics ?

The platform

Measuring application is already well covered :

- CPU, Memory, network

Measure the energy of an application:

- Sampling using RAPL (CPU instructions) [1]
  - Available on regular CPU (Intel, AMD)
  - Not available on ARM (specific instruction)
  - RAPL is not accurate (estimation)
- Some tool to simplify measurements in production, but not simple to use:
  - Scaphandre [3]
  - Greenspector [4]

- Energy measurement of GPU by design [2]
- Measuring energy consumption of containers is not simple

[1] KHAN, Kashif Nizam, HIRKI, Mikael, NIEMI, Tapio, et al. RAPL in Action: Experiences in Using RAPL for Power measurements. ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), 2018, vol. 3, no 2, p. 1-26.
[2] COPLIN, Jared et BURTSCHER, Martin. Effects of source-code optimizations on GPU performance and energy consumption. In : Proceedings of the 8th Workshop on General Purpose Processing using GPUs. 2015. p. 48-58.
[3] https://github.com/hubblo-org/scaphandre
[4] https://greenspector.com

The platform

A platform to evaluate the energy impact in the development process.

We integrate ENRICO to the CI/CD process

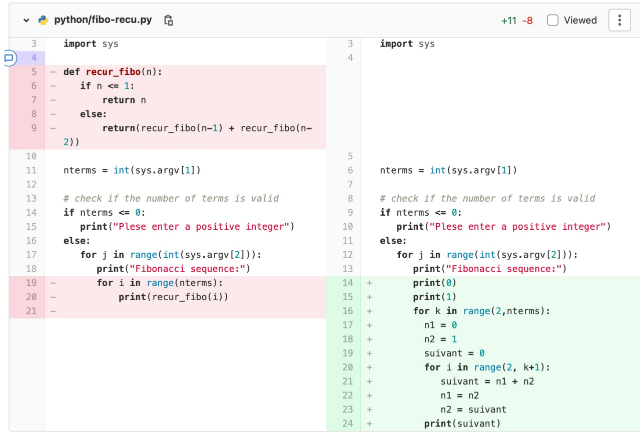
- Simple to integrate
- Simple to deploy
- Adapted to existing tools

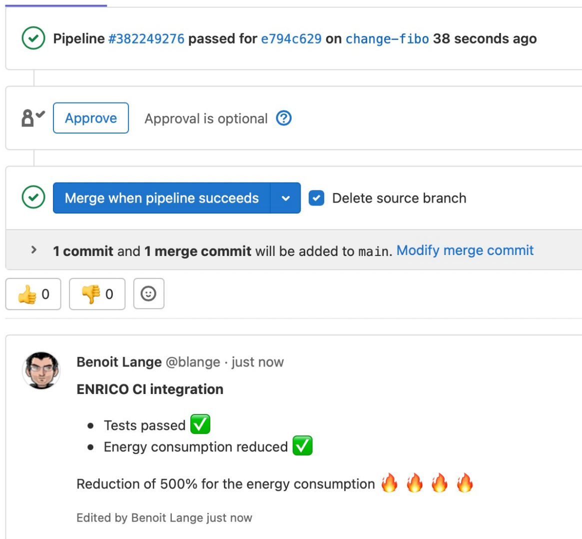
We extract metrics diff between two git commits

```
stages:
  - test
  - test-energy
  - build

test-all:
  image: gradle:jdk21
  stage: test
  script:
    - export GRADLE_USER_HOME=‘pwd‘/.gradle
    - ./gradlew  check test

test-enrico:
  image: gradle-enrico:jdk21
  stage: test-energy
  script:
    - export GRADLE_USER_HOME=‘pwd‘/.gradle
    - push-start-tag.sh
    - for i in {1..3};
    - do
    -     push-start-run.sh $i
    -     ./gradlew  test --tests org.acme.nbody.run1
    -     push-end-run.sh $i
    - done
    - push-stop-tag.sh
```



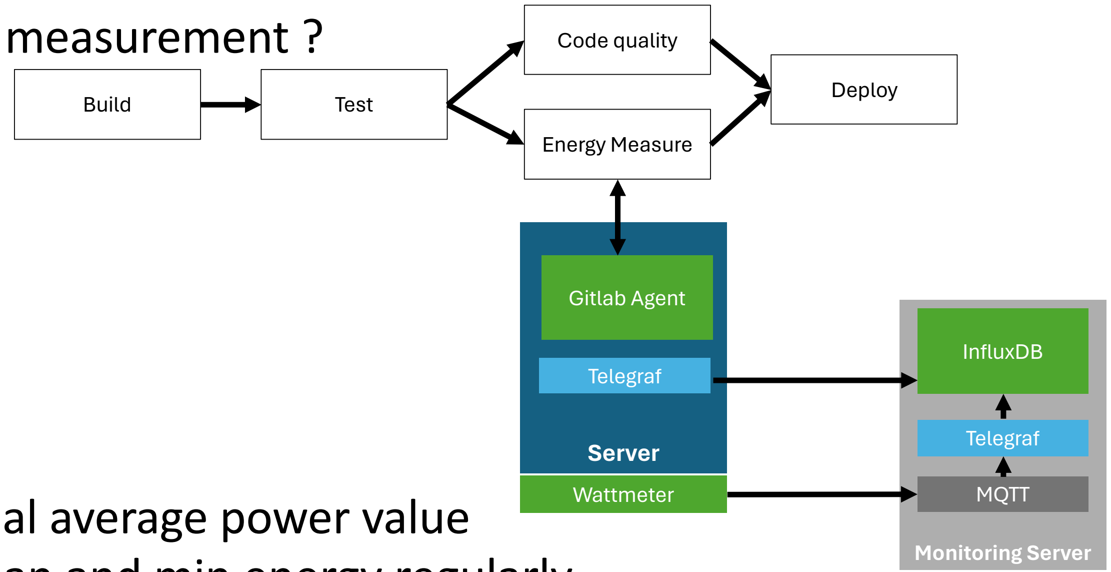


The platform

We use the test suite to evaluate the energy impact of a new feature or fix

How to get the best accuracy for energy measurement ?

- A dedicated node equipped with
  - A physical wattmeter
  - agent to collect system metrics
  - agent to collect RAPL metrics

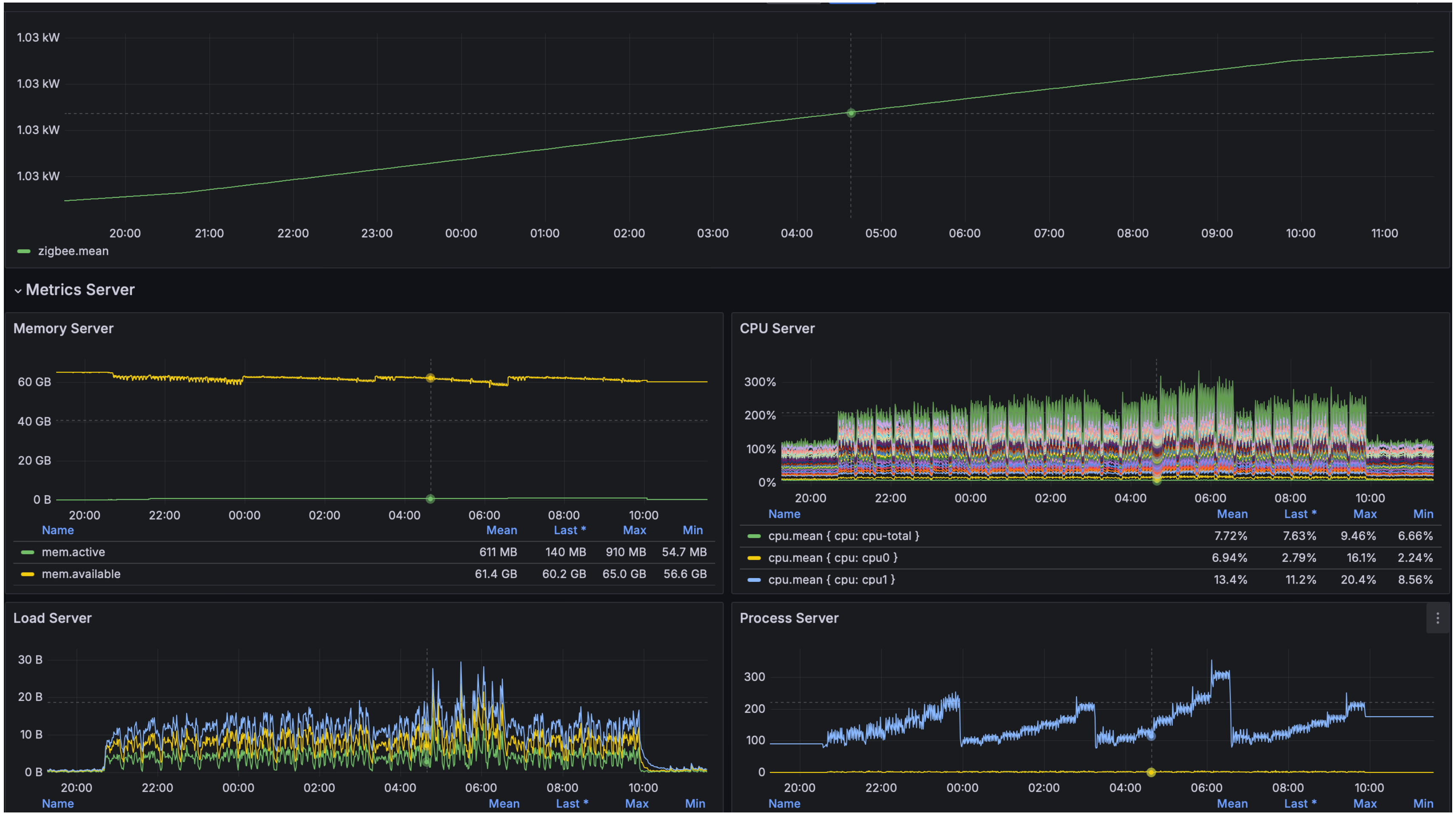


- Measures are sent to the TS DB

- Before each run we compute the initial average power value
  - For each run we collect max, mean and min energy regularly
- At the end we subtract initial average power value to max power (remove idle energy usage of the server)
  For the cumulated energy, we remove the theoretical slope to the measured one

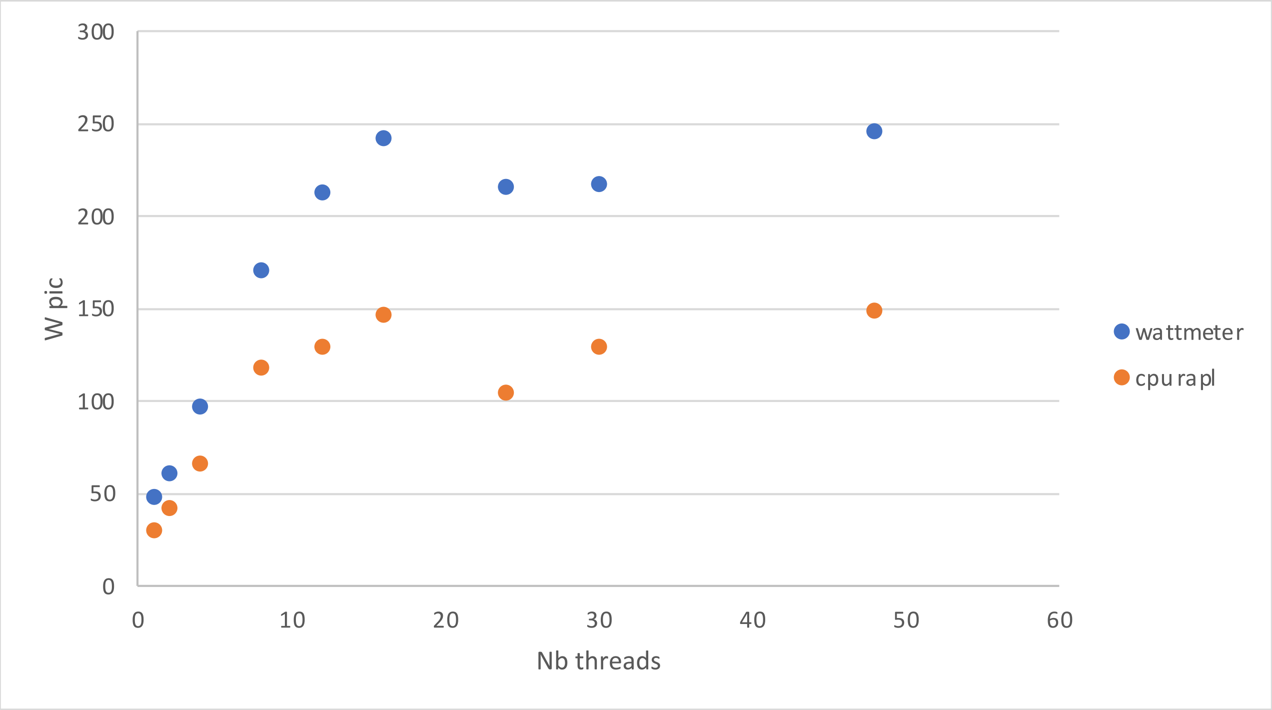
The platform

The platform



Results

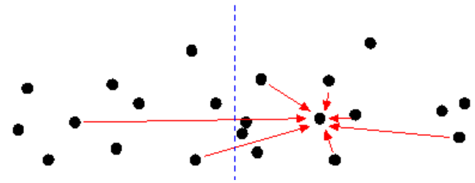
- Compare a physical wattmeter and RAPL on C++ code
- Gap between 1 thread is the hardware



- We use results of the wattmeter to preserve accuracy

Results

- A simple example to test our solution
- The n-body simulation For each particle, we compute gravitational interactions with other particles.
- The naive approach is in $O(n^2)$.
- **Highly parallelizable (suitable for GPU)**



Algorithm 1: The n-body algorithm.

```
Function compute_particle_force() is
    foreach i: body do
        compute_local_force(i, particles) ;

Function compute_local_force(i: body, particles) is
    foreach j in particles do
        if j ≠ i then
            d_sq = distance(i, j) ;
            ans[i].x += d_x * mass(i) / d_sq^3 ;
            ans[i].y += d_y * mass(i) / d_sq^3 ;
            ans[i].z += d_z * mass(i) / d_sq^3 ;
```

CHINCHILLA, Francisco, GAMBLIN, Todd, SOMMERVOLL, Morten, *et al.* Parallel n-body simulation using GPUs. *Department of Computer Science, University of North Carolina at Chapel Hill, http://gamma. cs. unc. edu/GPGP, Technical Report TR04-032*, 2004.

Results

- Naïve implementation without any optimization or complex feature
- We use Python (widely used in research)

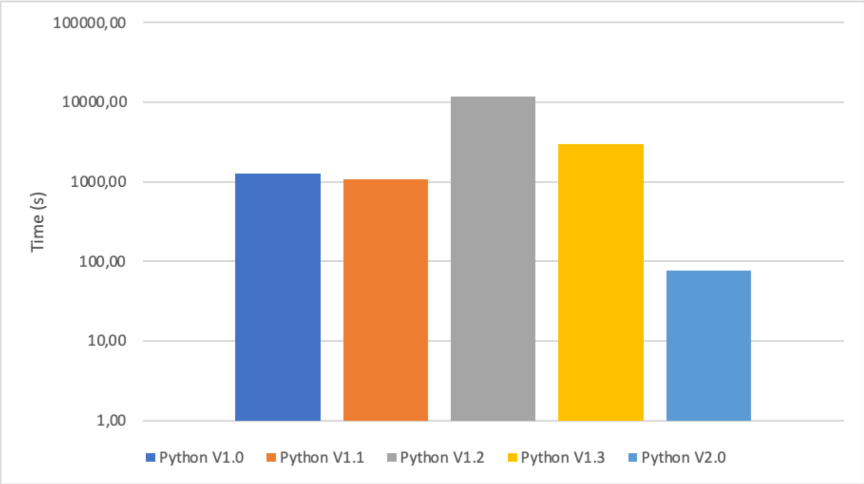
- Particles are composed of 6 dimensions:
  - X, Y, Z
  - Vx, Vy, Vz

- Optimization based on data structure or library

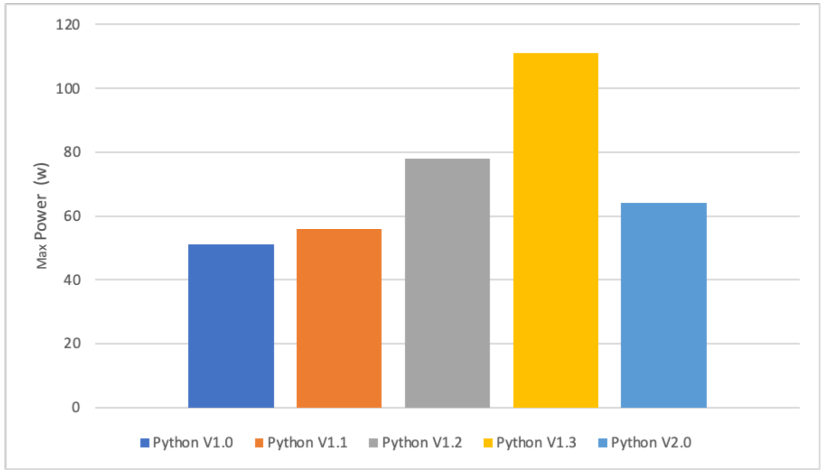
Results

- **We propose 4+1 iterations of the implementation**
  - V 1.0: particles are store in a one-dimensional vector where each dimension is contiguous to each other
  - V 1.1: particles are stored in a one-dimensional vector where each dimension is stored into a blob
  - V 1.2: particles are stored in multiple arrays that are stored within one array
  - V 1.3: particles are stored in a 2D array
  - V 2.0: numpy implementation
- **The test run a simulation, based on an initial step for the simulation with 10k particles and running for 100 steps**

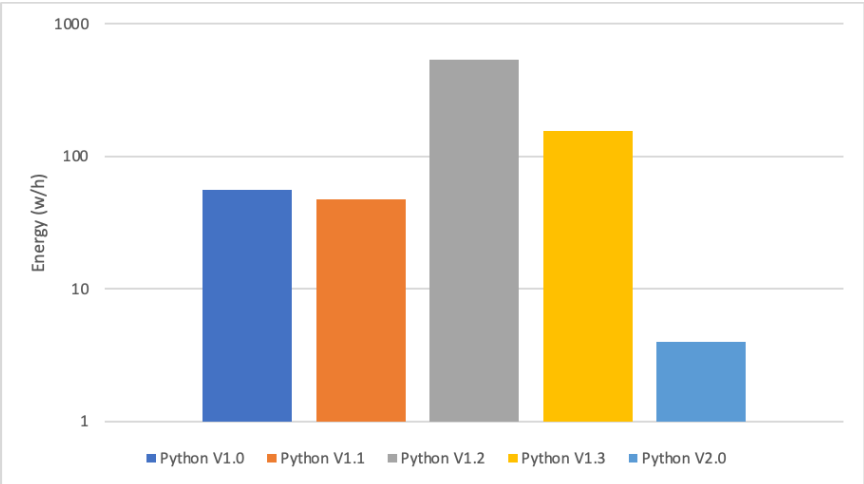
Results



100000,00
10000,00
1000,00
100,00
10,00
1,00
Time (s)
Python V1.0
Python V1.1
Python V1.2
Python V1.3
Python V2.0



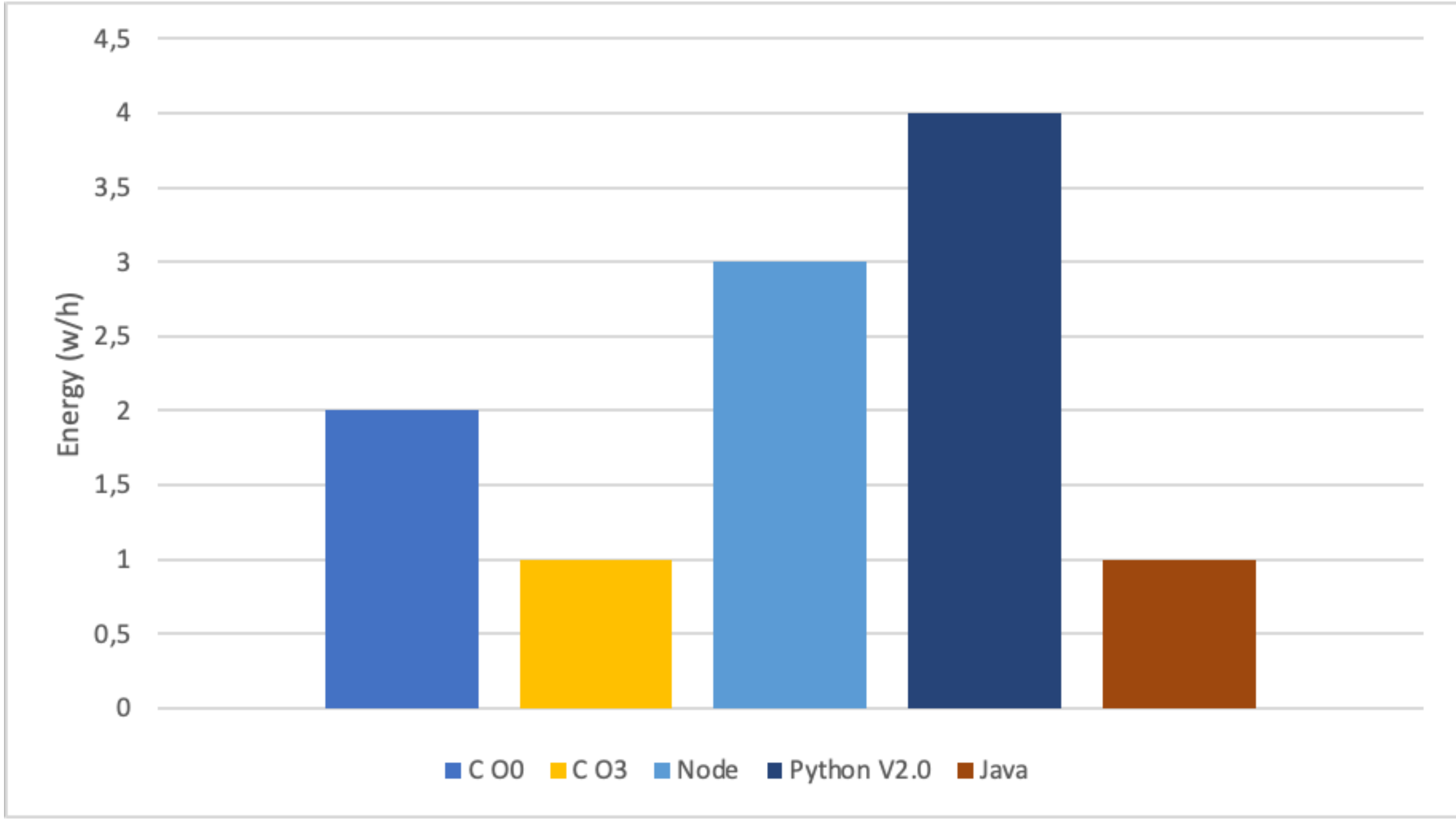
120
100
80
60
40
20
0
Max Power (w)
Python V1.0
Python V1.1
Python V1.2
Python V1.3
Python V2.0



1000
100
10
1
Energy (w/h)
Python V1.0
Python V1.1
Python V1.2
Python V1.3
Python V2.0

Results

- **Is it worth to move to another language ?**



Conclusions

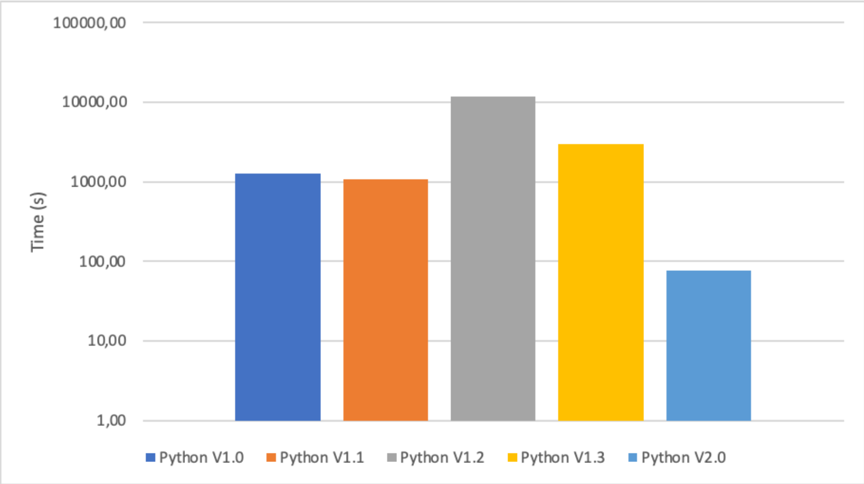
- **We propose a new CI/CD step to monitor energy impact of a new feature, bug fix based on test suit**
- **We use RAPL and a wattmeter to monitor the energy impact**
  - Wattmeter is more accurate for multi threading
- **We measure optimization of a python application and show the impact of code evolution**
- **We measure this version of python app with other language**
- **In a future, we want to provide insight of code optimization based on energy impact**
- **Measure code optimization proposed by sonar**
- **Give to developer in real time energy impact of code**

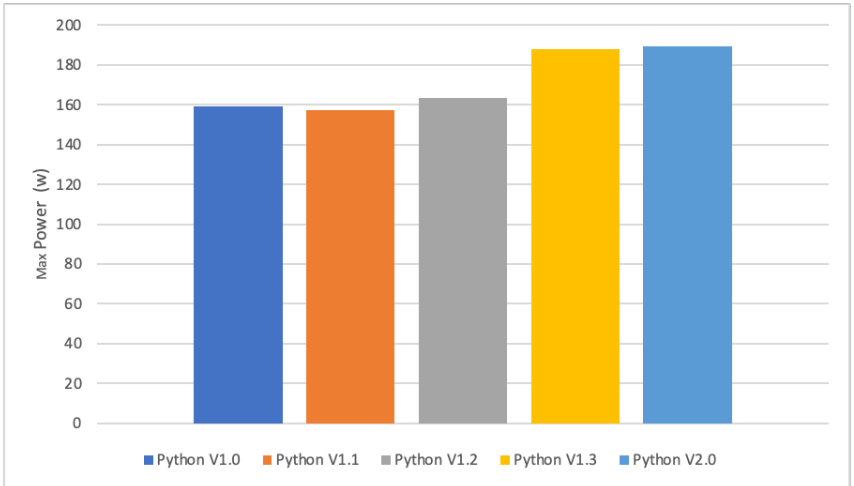
Thank you for the Attention

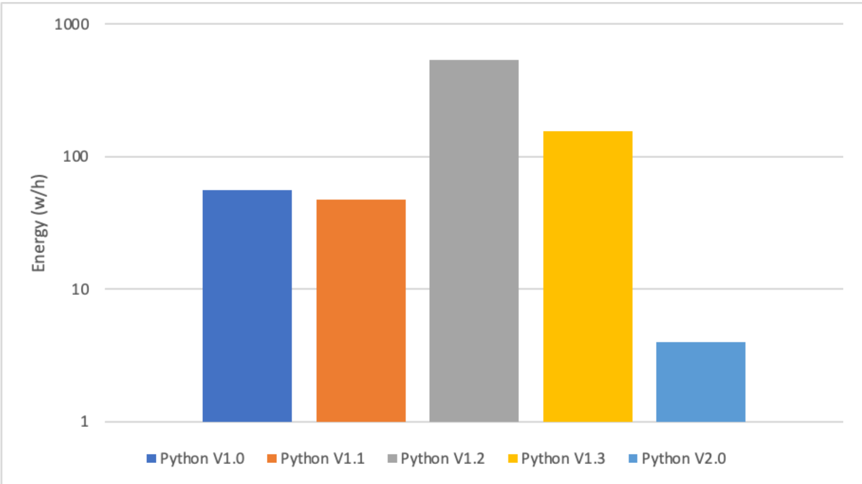
Benoit Lange
Benoit.lange@inria.fr

24 March 2025

Results


100000,00
10000,00
1000,00
100,00
10,00
1,00
Time (s)
Python V1.0
Python V1.1
Python V1.2
Python V1.3
Python V2.0


200
180
160
140
120
100
80
60
40
20
0
Max Power (w)
Python V1.0
Python V1.1
Python V1.2
Python V1.3
Python V2.0


1000
100
10
1
Energy (w/h)
Python V1.0
Python V1.1
Python V1.2
Python V1.3
Python V2.0