

When Faster Isn't Greener: The Hidden Costs of LLM-based code optimization

Presented by:
Tristan Coignon

Empirical Evaluation of the Energy Impact of Large Language Models for Code Generation and Optimization

Presented by:
Tristan Coignon

Empirical Evaluation of the Energy Impact of Large Language Models for Code Generation and Optimization

Presented by:
Tristan Coignon

Examiners:

- Jean-Christophe Routier
- Romain Robbes
- Aurélie Bugeau

Advisors:

- Romain Rouvoy
- Clément Quinton

Rapporteur.euses:

- Anne-Laure Ligozat
- Laurent Lefevre



Environmental impact of the digital sector

- **4.4 %** of France's total carbon footprint
- **11 %** of France's electricity use
- **117 Mt** resources consumed yearly → **1.7 t/person**

GHG emissions may triple by 2050 if nothing changes

Source: ADEME 2025

Where do these impacts come from?

By equipment type

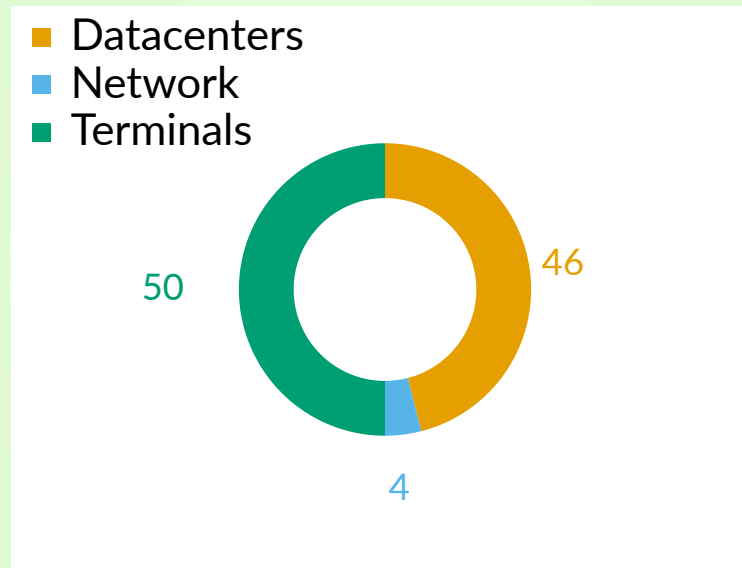


Fig. I.A. GHG Emissions by type of equipment

Where do these impacts come from?

By equipment type

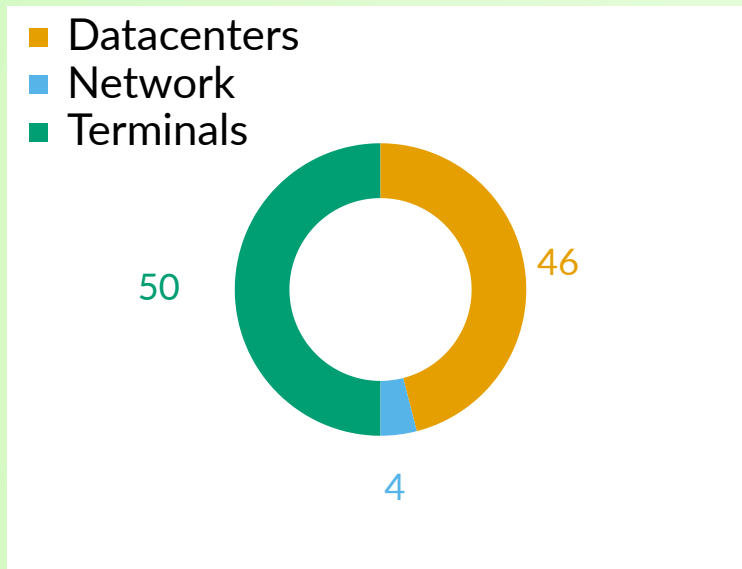


Fig. I.A. GHG Emissions by type of equipment.

By lifecycle phase

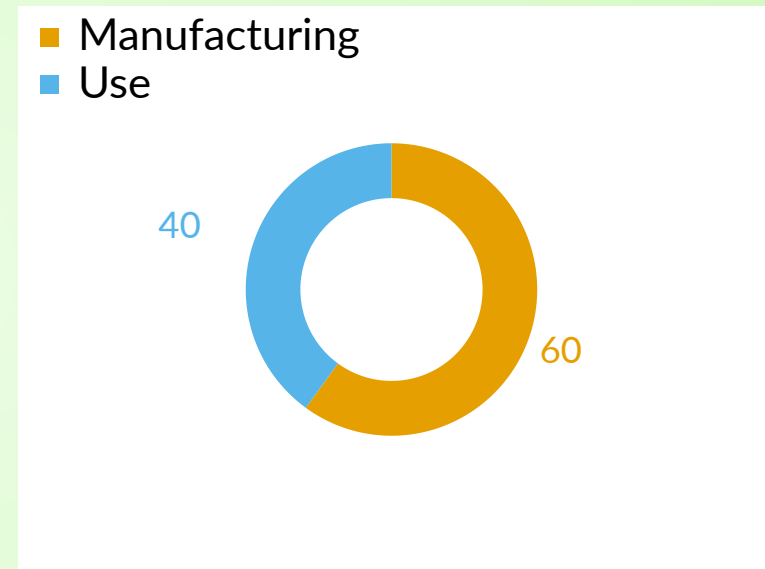


Fig. I.B. GHG Emissions by lifecycle phase.

Software usage is driving these impacts

- Audio and video streaming
- Crypto currencies
- Artificial Intelligence

Generative AI amplifies digital demand

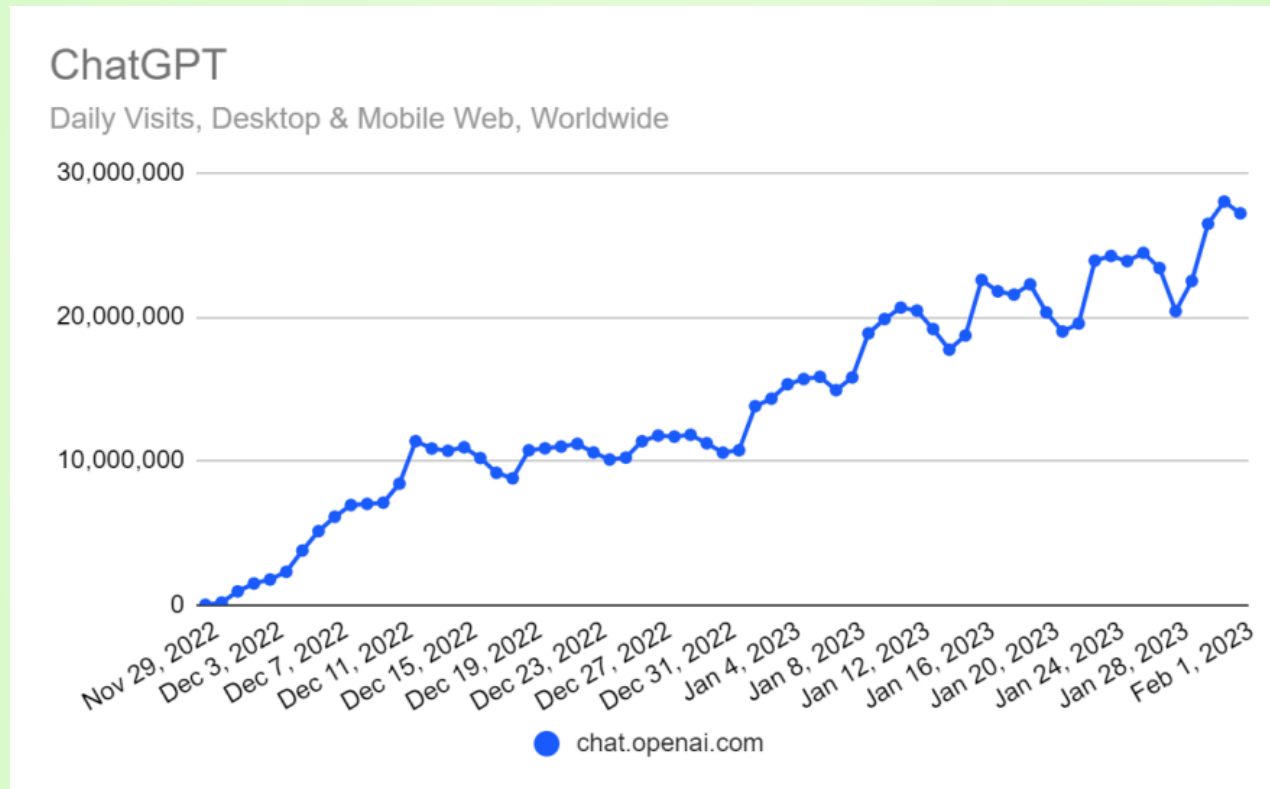


Fig. I.C. ChatGPT's userbase growth.

Generative AI amplifies digital demand

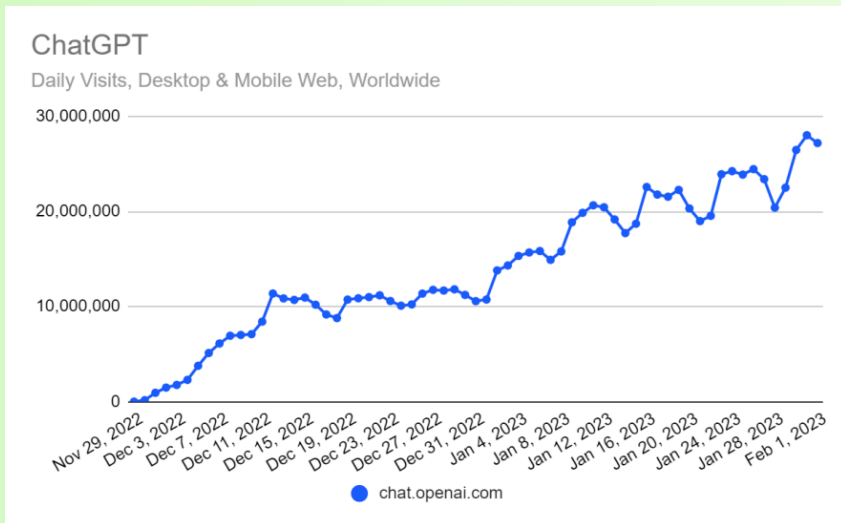


Fig. I.C. ChatGPT's userbase growth.

- Over **100 millions users** in two months
- Nvidia → most valuable company in the world
- Microsoft builds nuclear power plants

Large Language Models need a lot of energy

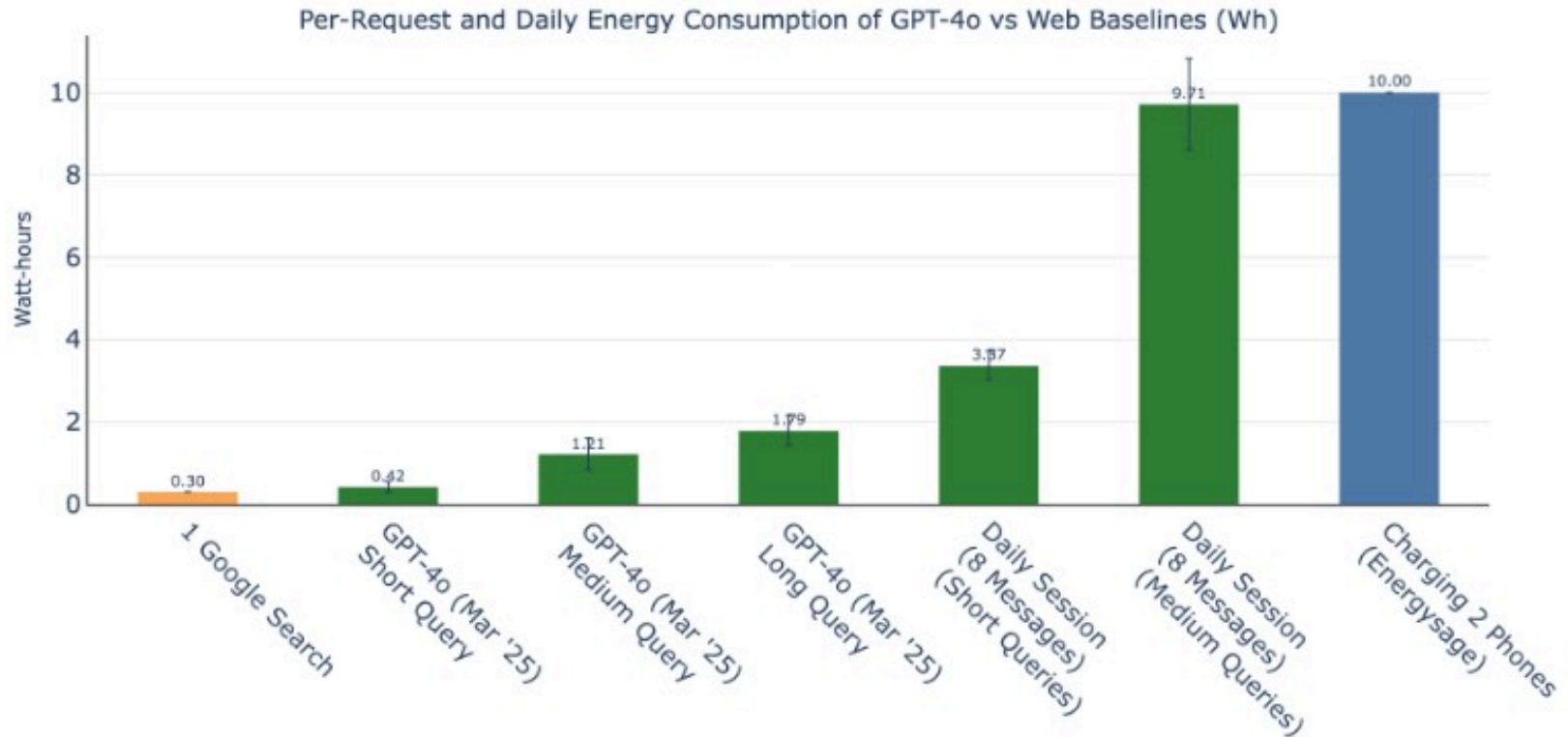


Fig. I.D. Energy consumption of ChatGPT.

Source: *How Hungry is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference*

Large Language Models need a lot of energy

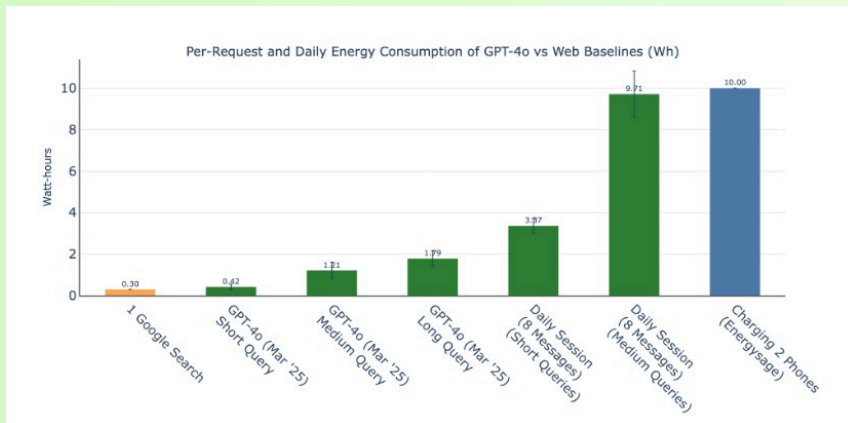


Fig. I.D. Energy consumption of ChatGPT.

- Millions of daily inferences
- Integration into IDEs, chatbots, and search engines

Source: *How Hungry is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference*

The portion of software produced with the help of Large Language Models is increasing

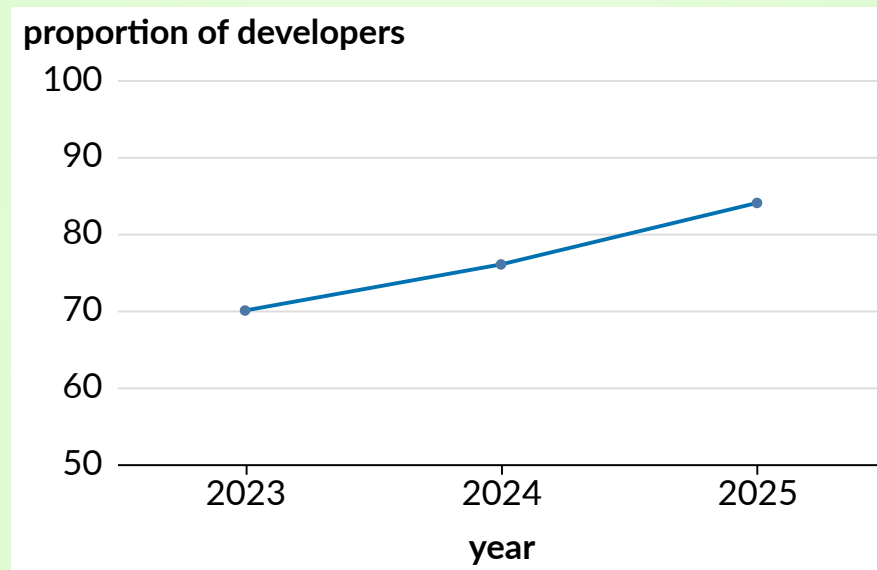


Fig. I.E. Amount of developers using or planning to use AI tools.

Source: StackOverflow Surveys [2023-2025]

Problem statement

What are the negative and positive energy impacts of code LLMs?

What are the negative and positive energy impacts of code LLMs?

1. How efficient is the code generated by LLMs?
2. How much energy is used when using code assistants like GitHub Copilot?
3. Is optimizing code using LLMs energetically worth it?

Plan

1. **Efficiency of LLM-generated code**
2. Energy consumption of a code assistant
3. Optimizing software with LLMs
4. Conclusion

An increasing number of developers are using LLMs to write code for them

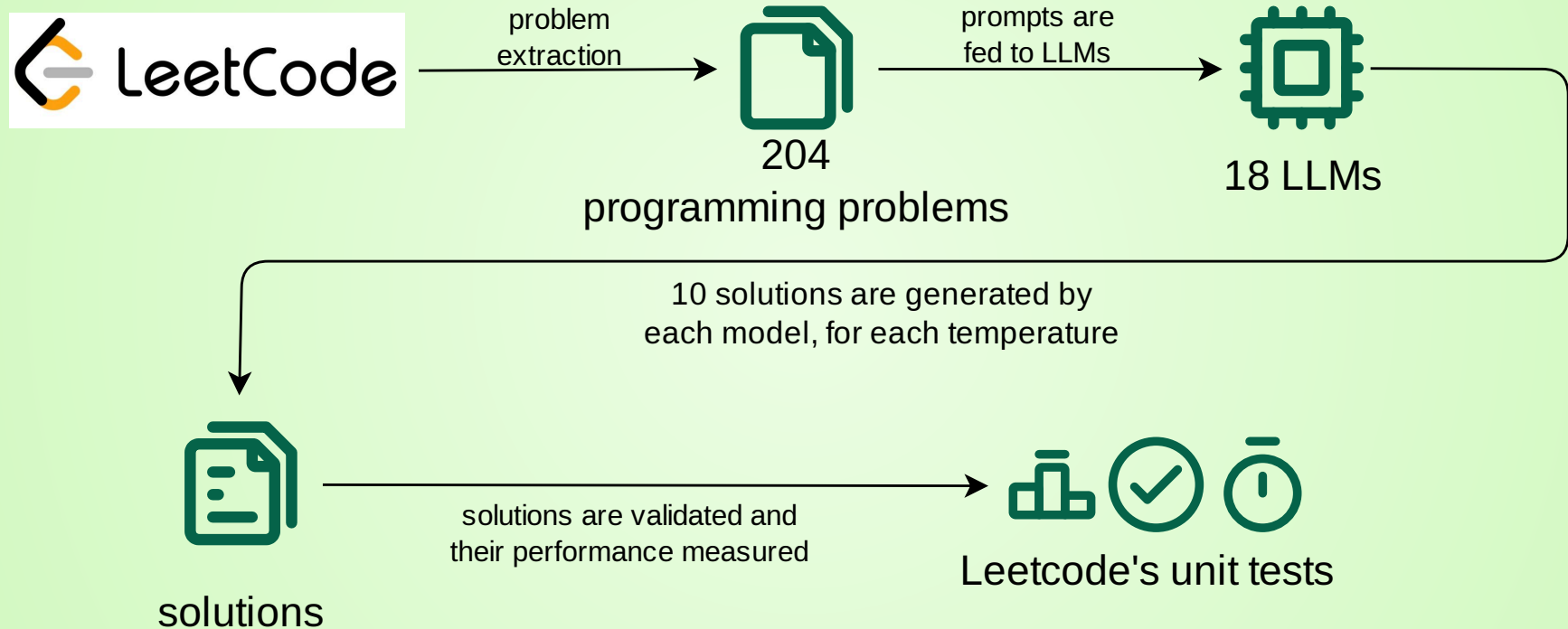
84% of respondents are using or planning to use AI tools [...], an increase over last year (76%).

Source: StackOverflow 2025 Developer Survey

Research Question

How efficient is the code generated by LLMs?

Method



Results

Results

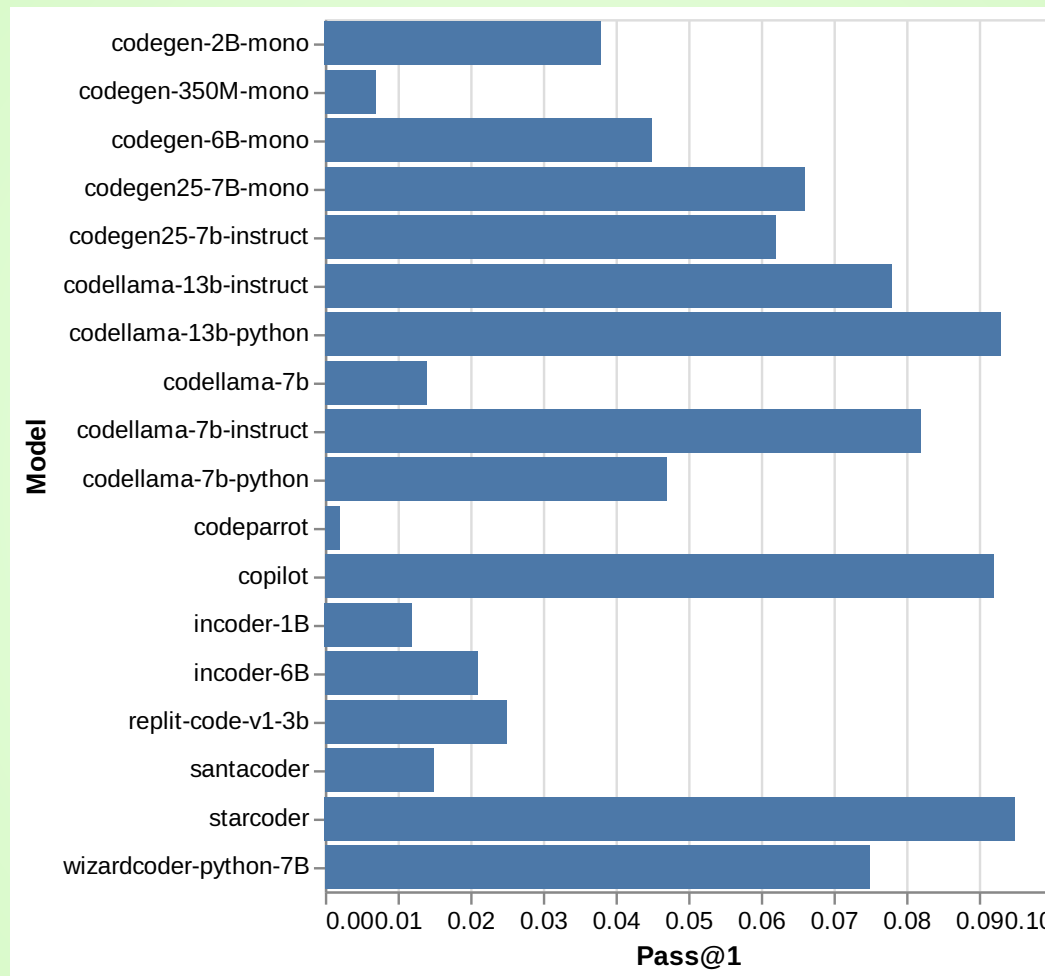


Fig. II.A. Correctness of the solutions depending on the model.

Results

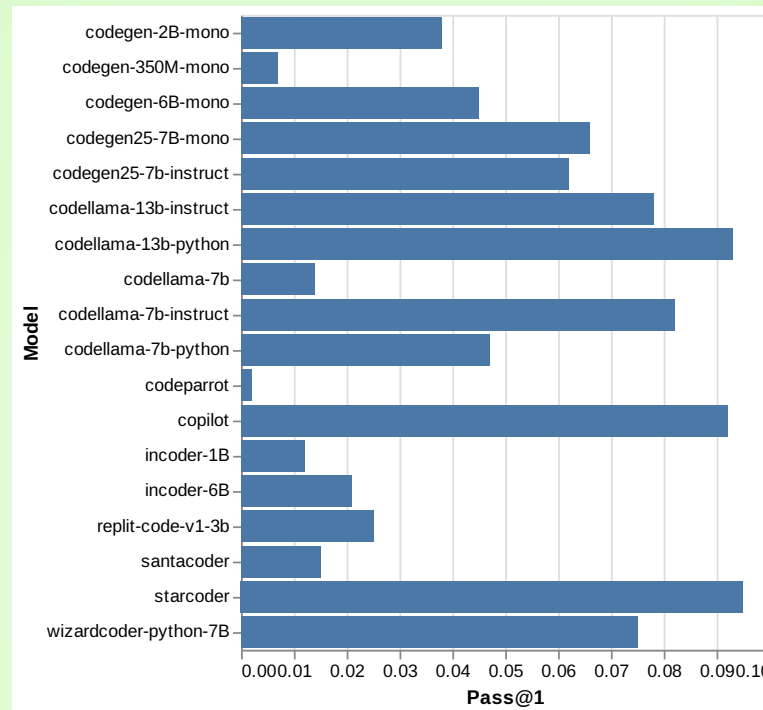


Fig. II.A. Correctness of the solutions depending on the model.

The correctness of the generated solution was strongly dependent on the model.

Results

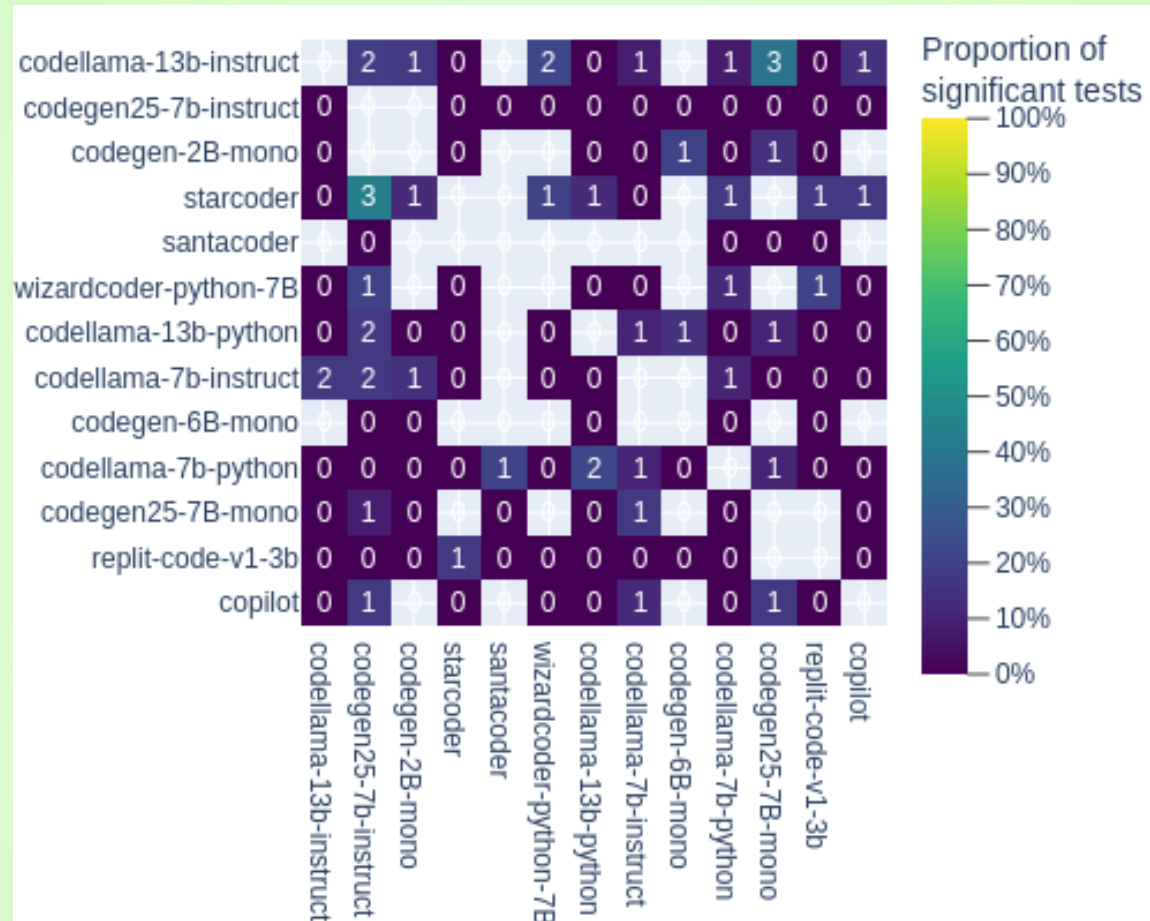


Fig. II.B. Number of problems where a model (row) produces more efficient code than another (column); Mann-Whitney test; $\alpha=0.05$.

Results

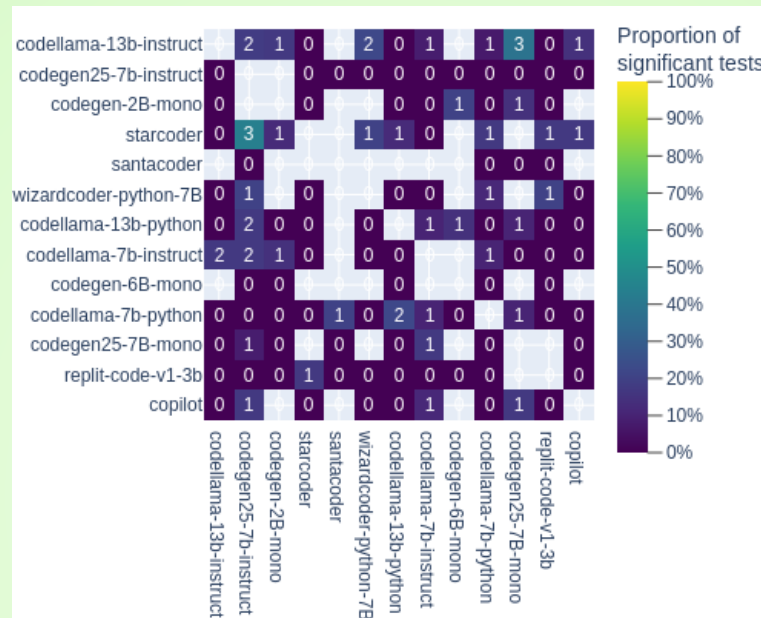


Fig. II.B. Number of problems where a model (row) produces more efficient than another (column); Mann-Whitney test; $\alpha=0.05$.

The efficiency of LLM-generated code is largely similar across different models, regardless of their size or training data.

Results

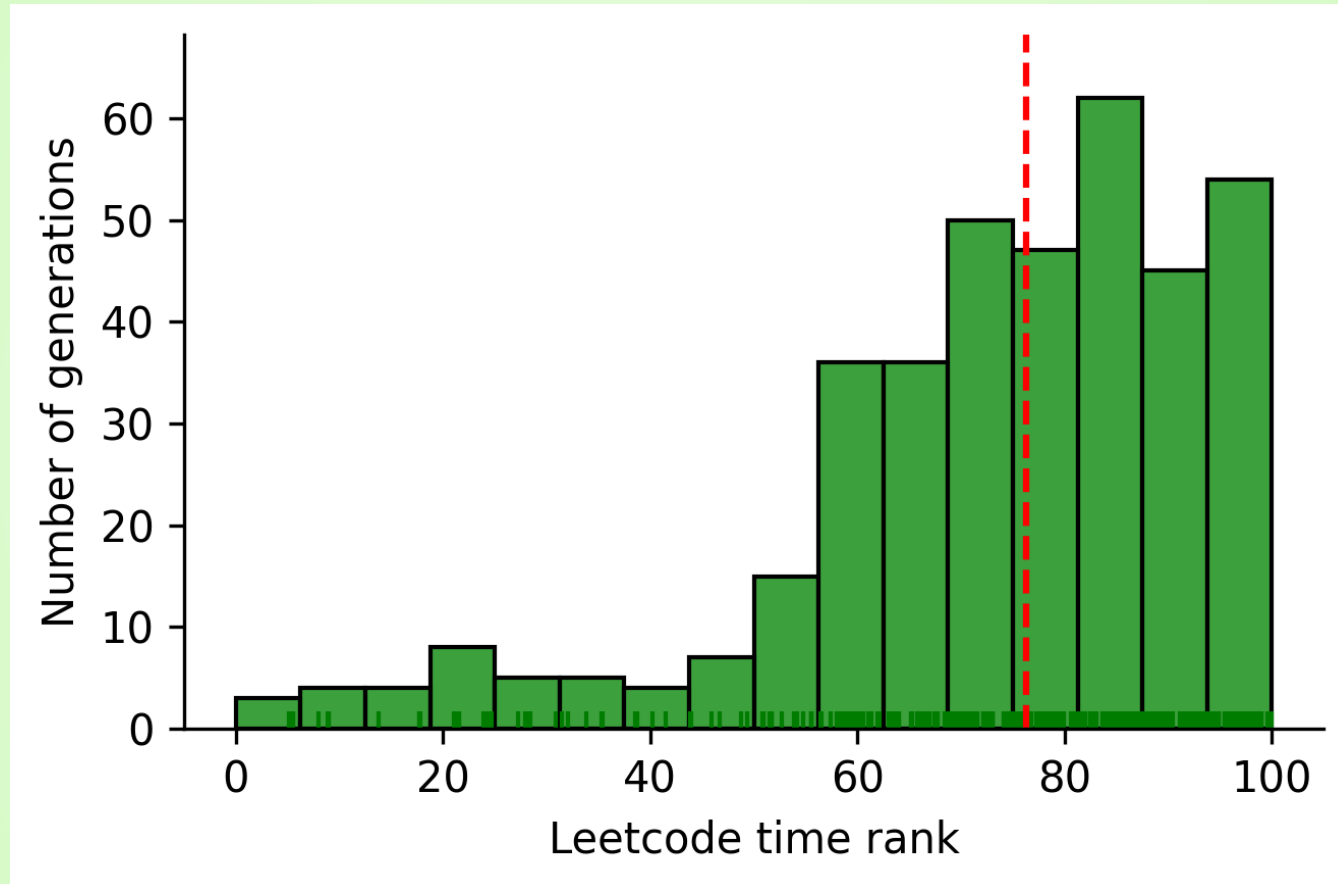


Fig II.C. Efficiency of the generated code compared to humans (Higher is better). Median indicated in red.

Results

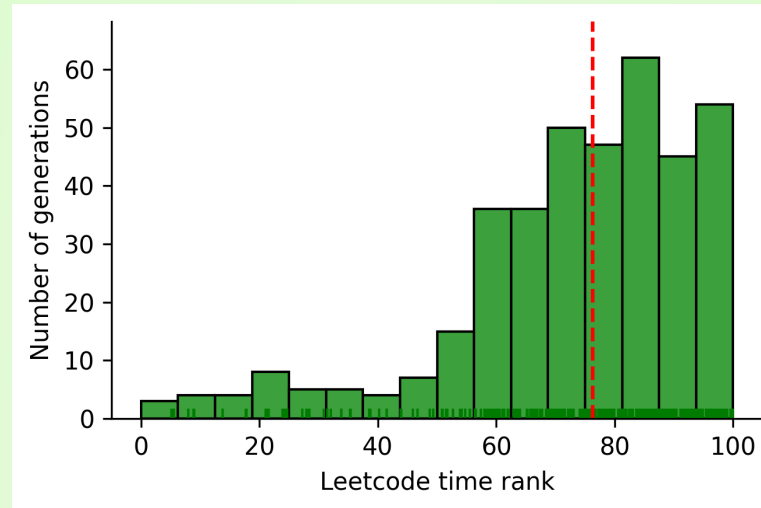


Fig II.C. Efficiency of the generated code compared to humans (Higher is better). Median indicated in red.

LLM-generated solutions are on average faster than 73% of the other solutions on Leetcode.

Takeaways

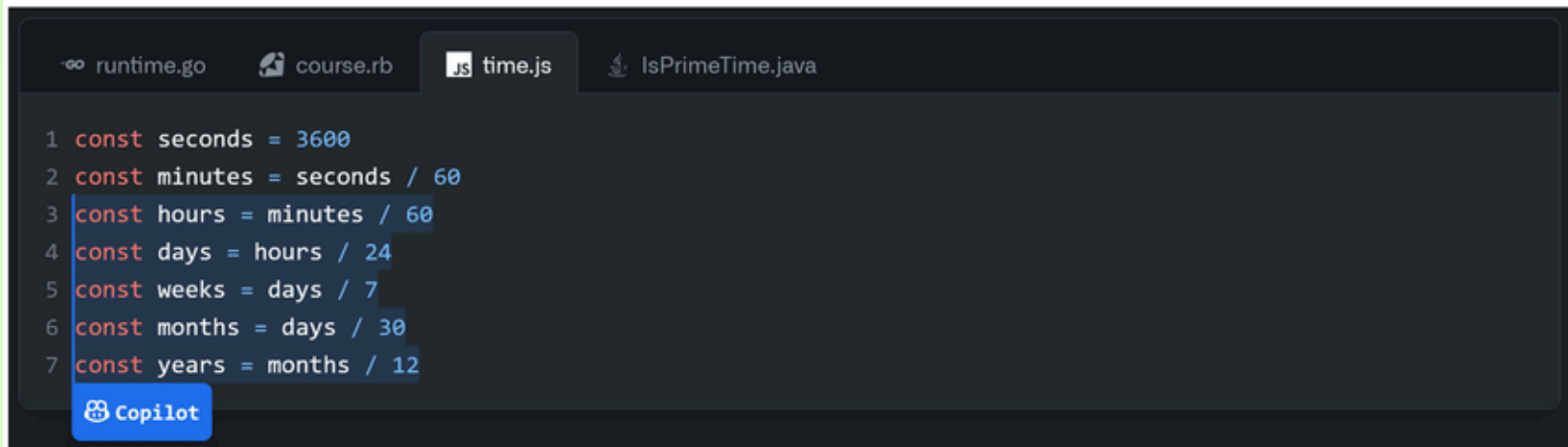
- LLMs produce code with comparable efficiency, **irrespective of the adopted LLM.**
- LLMs are capable of generating code that is, on average, **more efficient than the code written by humans.**

[1] Coignon, T., Quinton, C., & Rouvoy, R. (2024). A Performance Study of LLM-Generated Code on Leetcode. Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (EASE), 79–89.

Plan

1. Efficiency of LLM-generated code
2. **Energy consumption of a code assistant**
3. Optimizing software with LLMs
4. Conclusion

GitHub Copilot



The screenshot shows a code editor with four tabs: runtime.go, course.rb, time.js (active), and IsPrimeTime.java. The time.js file contains the following code:

```
1 const seconds = 3600
2 const minutes = seconds / 60
3 const hours = minutes / 60
4 const days = hours / 24
5 const weeks = days / 7
6 const months = days / 30
7 const years = months / 12
```

GitHub Copilot suggestions are shown as blue overlays on lines 3, 4, 5, 6, and 7. A Copilot icon is visible in the bottom left corner of the editor.



The screenshot shows the Visual Studio Code interface with a file named max_sum_slice.py. The code is as follows:

```
1 def max_sum_slice(xs):
2     max_ending = max_so_far = 0
3     for x in xs:
4         max_ending = max(0, max_ending + x)
5         max_so_far = max(max_so_far, max_ending)
6     return max_so_far
```

GitHub Copilot suggestions are shown as blue overlays on lines 3, 4, 5, and 6. A Copilot icon is visible in the bottom left corner of the editor.

Research Question

How much energy does one developer consume when using a code assistant like GitHub Copilot?

Method

20 participants developing
for one hour

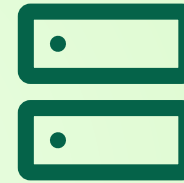


uses



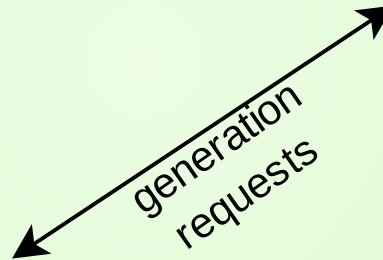
GitHub Copilot

GitHub Copilot's
inference server



Microsoft
Azure

generation
requests



Method

20 participants developing
for one hour



uses



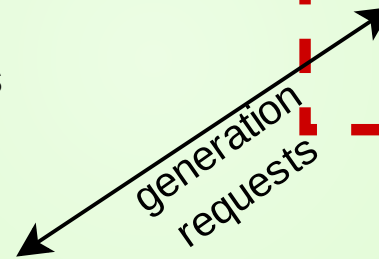
GitHub Copilot

GitHub Copilot's
inference server



Microsoft
Azure

generation
requests





1393757298

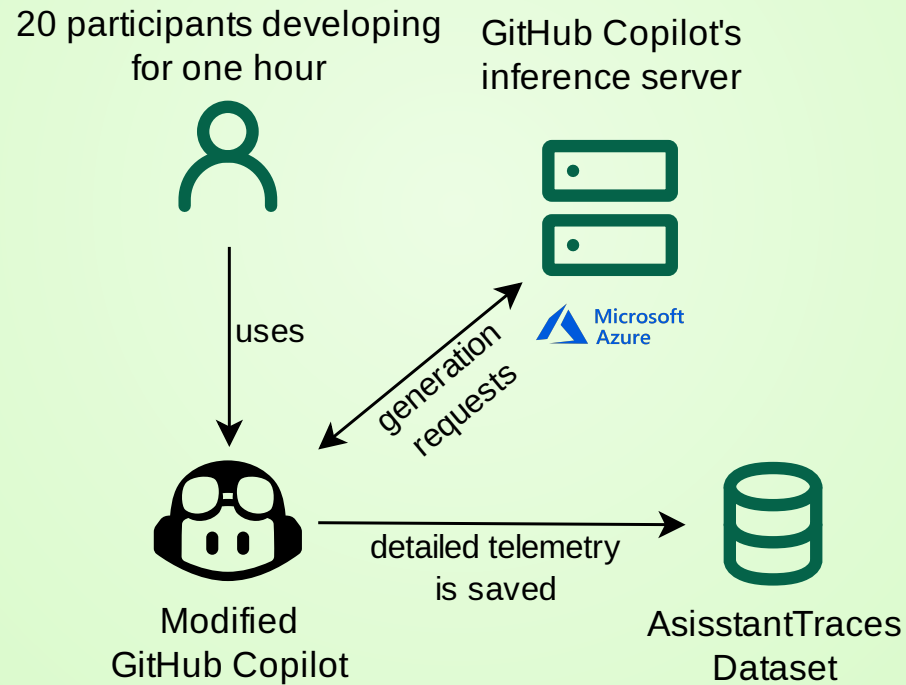


gettyimages®
Credit: boonchai wedmakawand

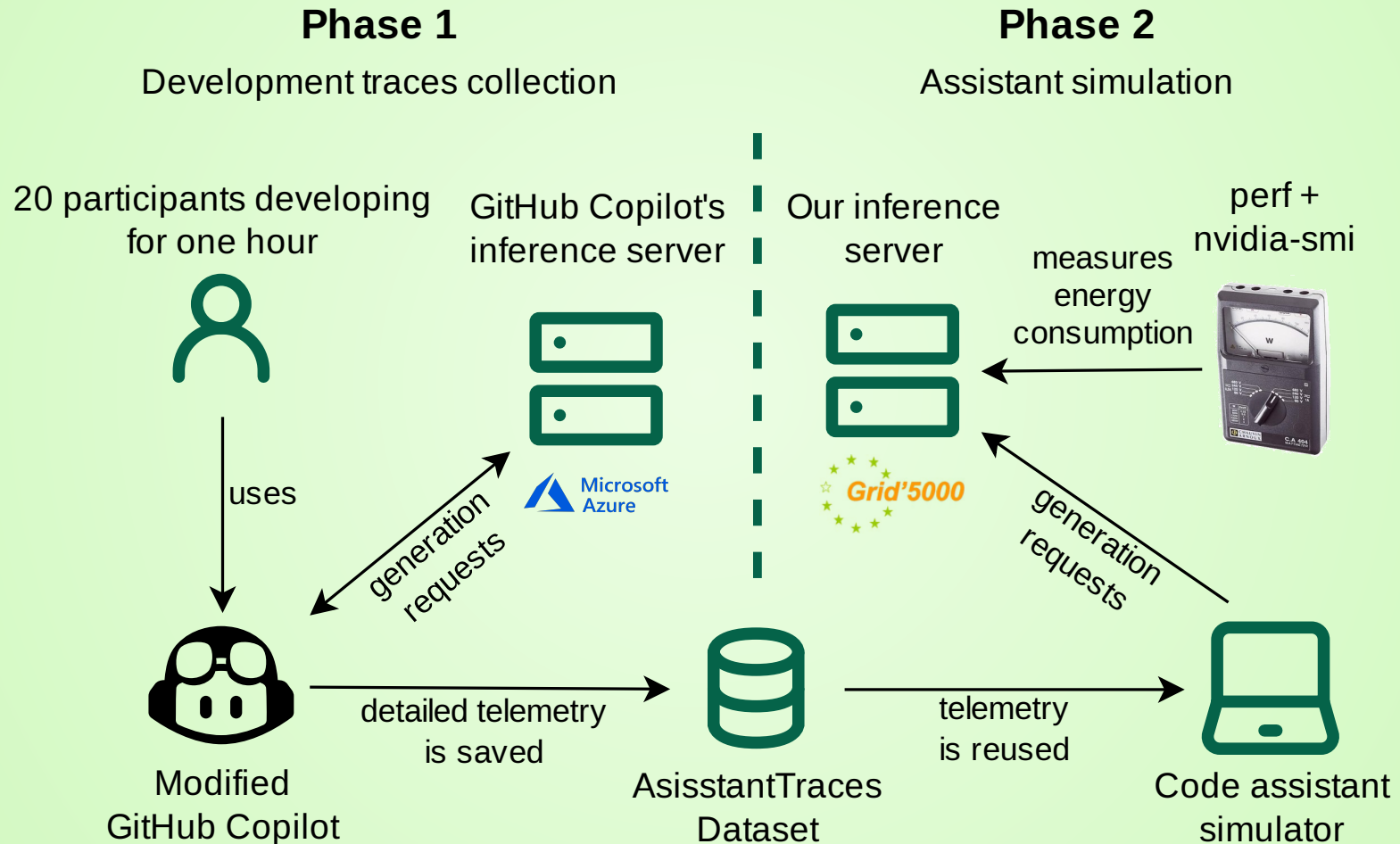
Method

Phase 1

Development traces collection



Method



Configuration space

- Number of developers being simulated (1-500)
- Model (3 LLMs)
- Quantization (3 methods)
- Number of GPUs (1-4)
- Request streaming (on/off)

Results

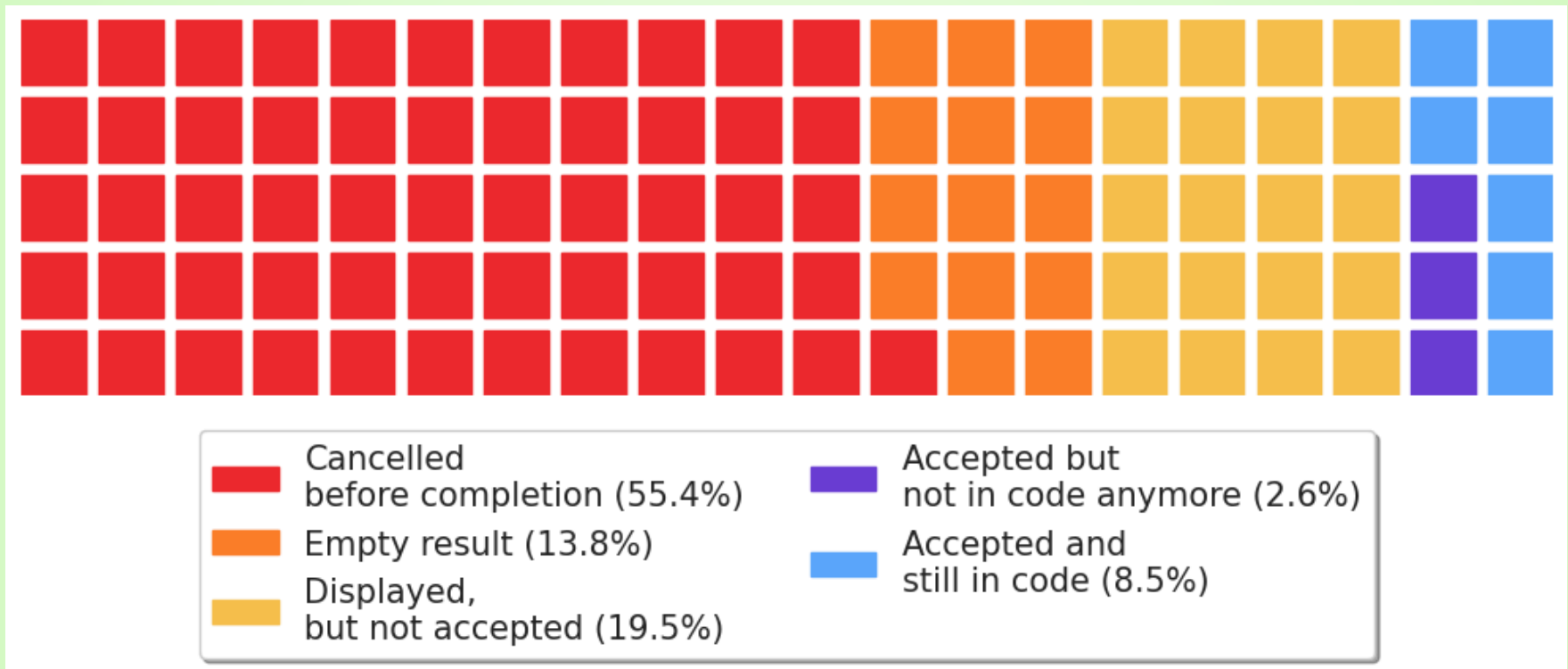


Fig. III.A. Percentage of requests sent by GitHub Copilot depending on their completion state.

Results

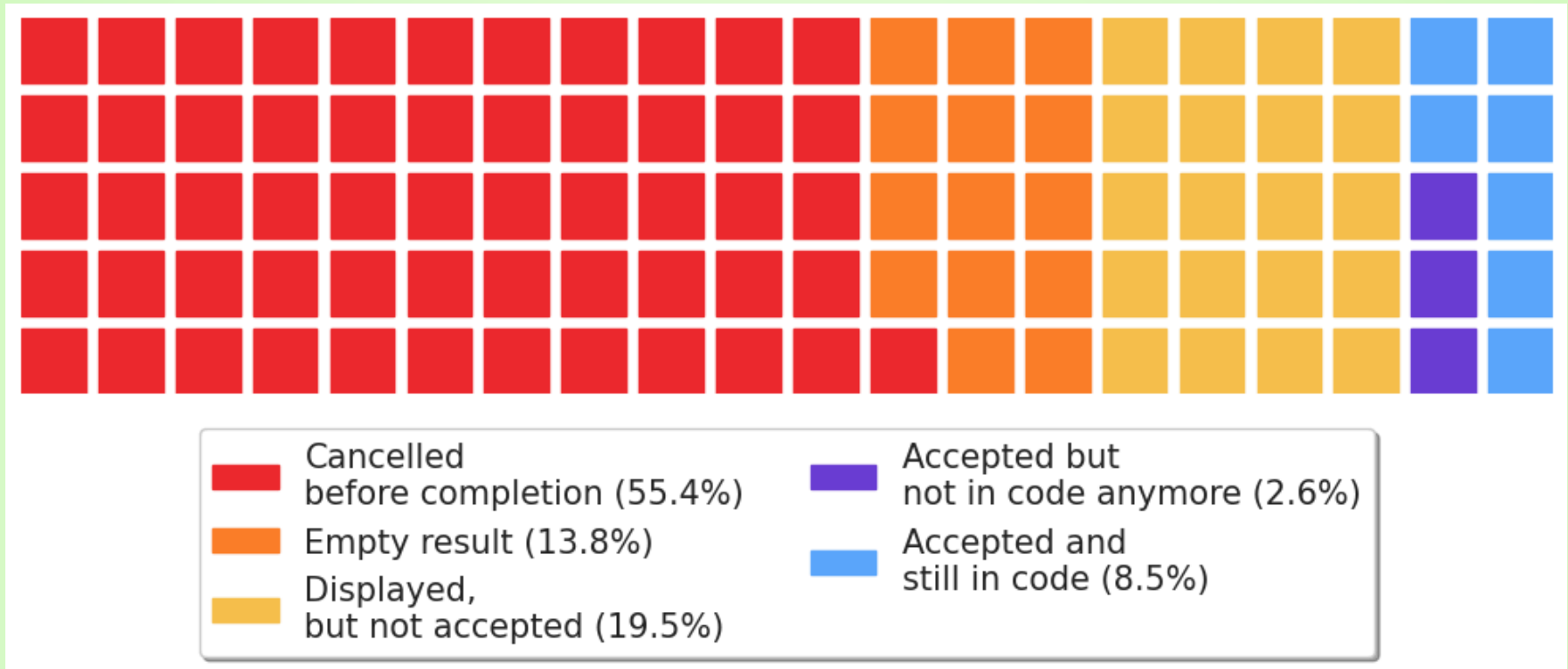


Fig. III.A. Percentage of requests sent by GitHub Copilot depending on their completion state.

Only 10% of the generations made by GitHub Copilot are used by the developers.

Results

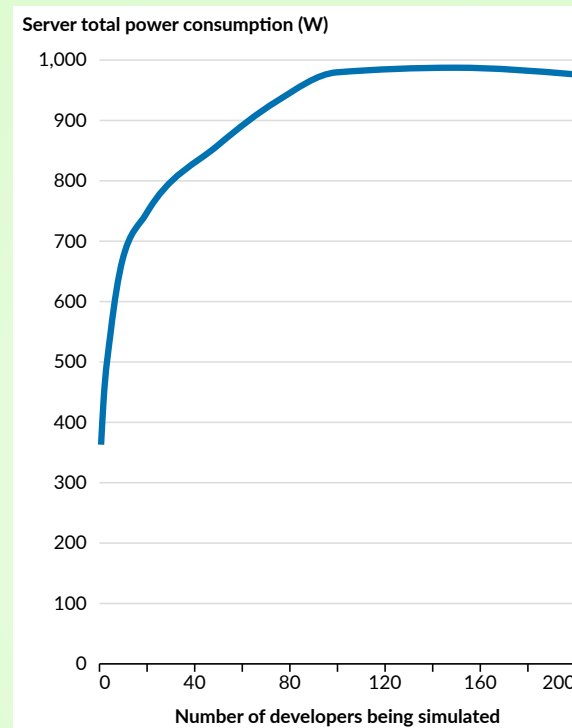


Fig. III.B. Energy consumption of the server.

The energy consumption **does not scale linearly** with the number of developers.

Results

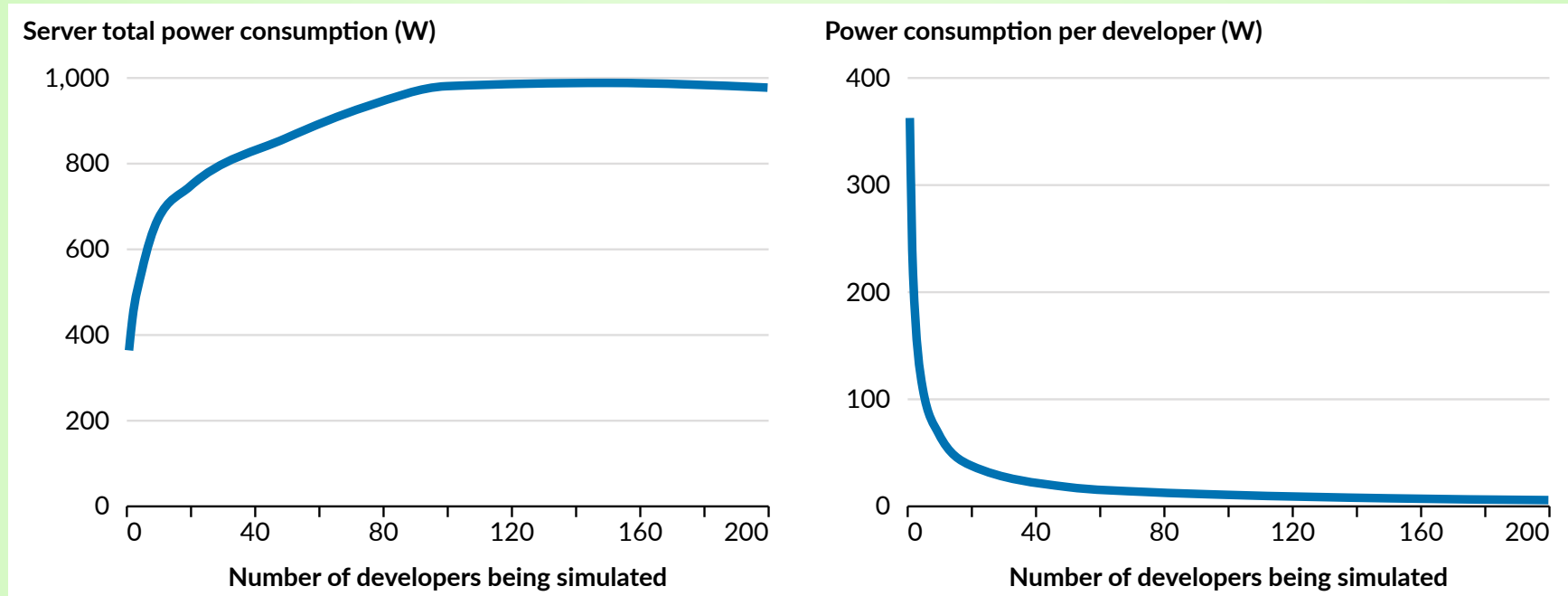


Fig. III.B. Energy consumption of the server + Energy consumption by developer.

More developers = Less energy consumption per developer.

Results

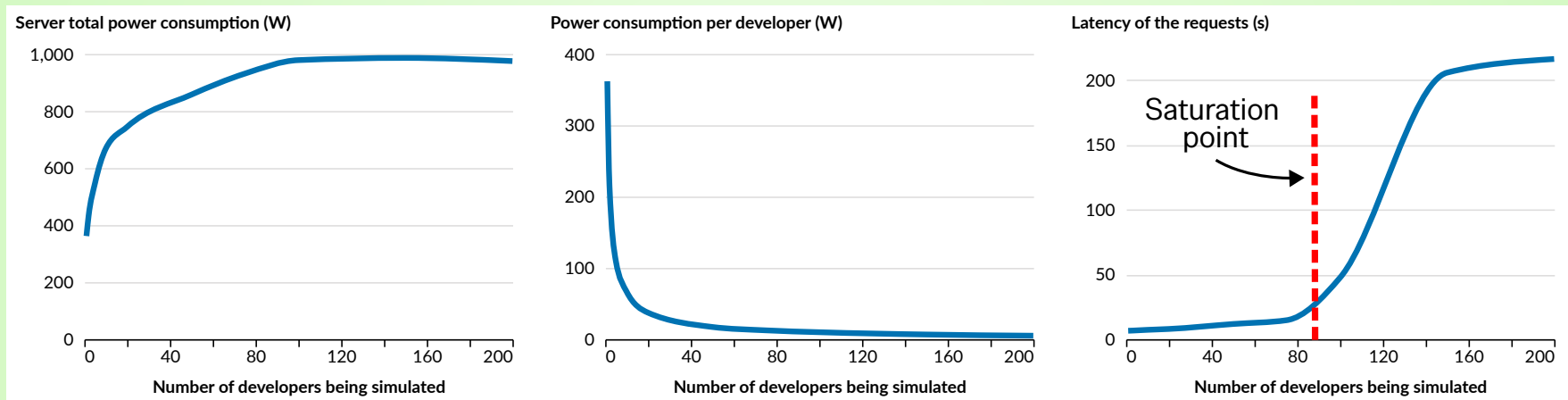


Fig. III.B. Energy consumption of the server + Energy consumption by developer + Latency of the requests.

The server **saturates** after too many developers are added.

Results

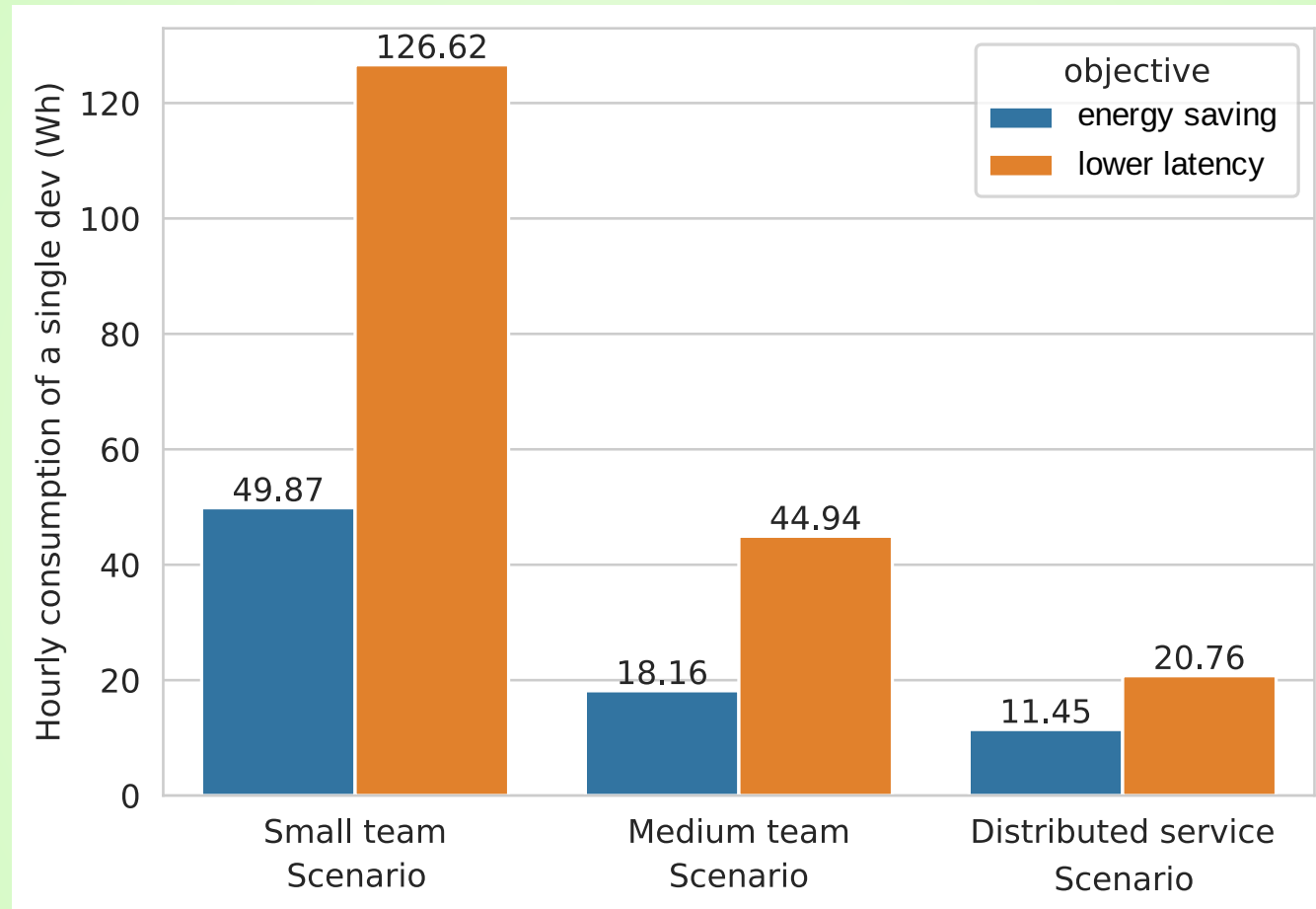


Fig. III.C. Energy consumption of a developer depending on the number of users on the inference server and the objective of the provider.

Results

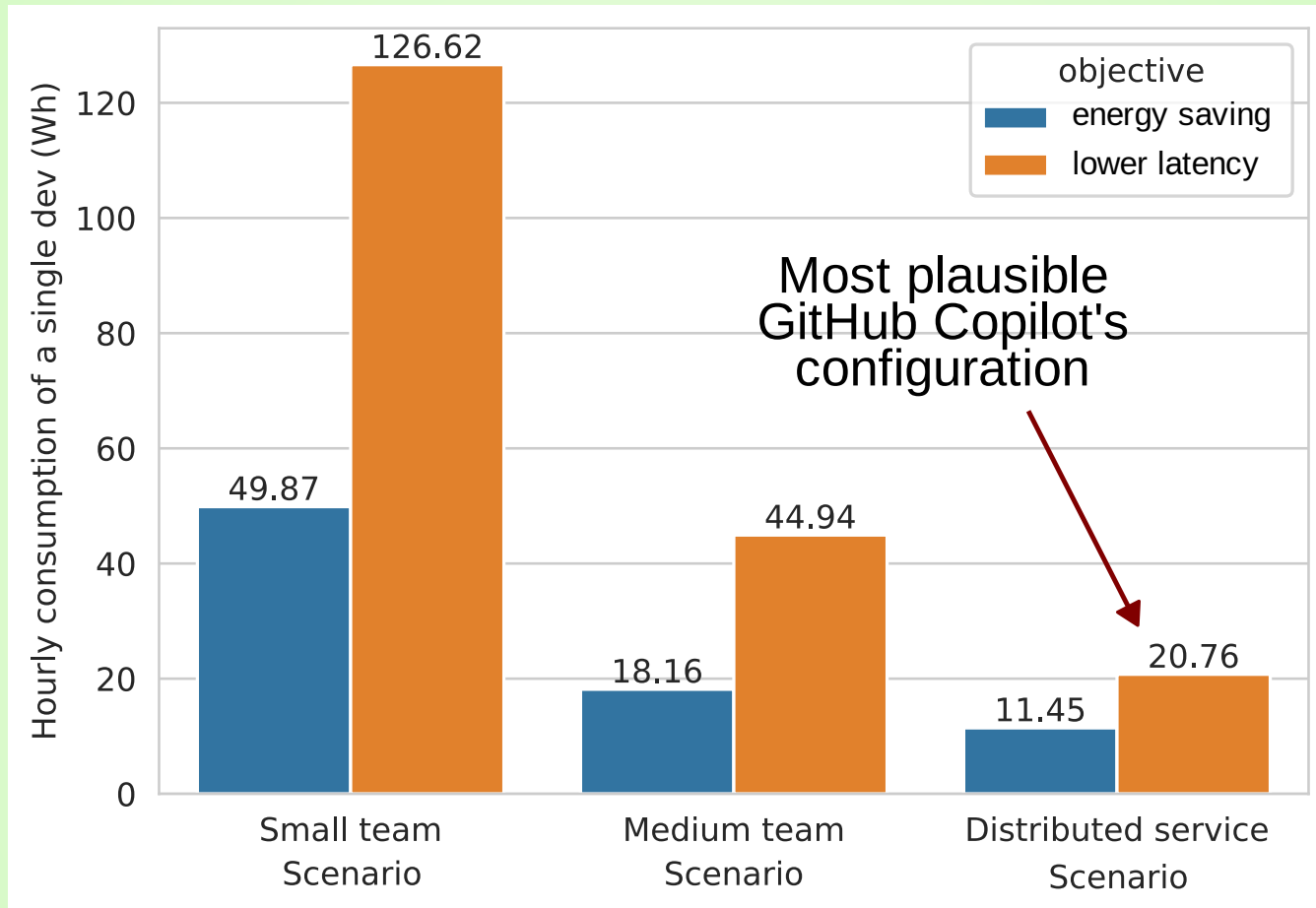


Fig. III.C. Energy consumption of a developer depending on the number of users on the inference server and the objective of the provider.

Results

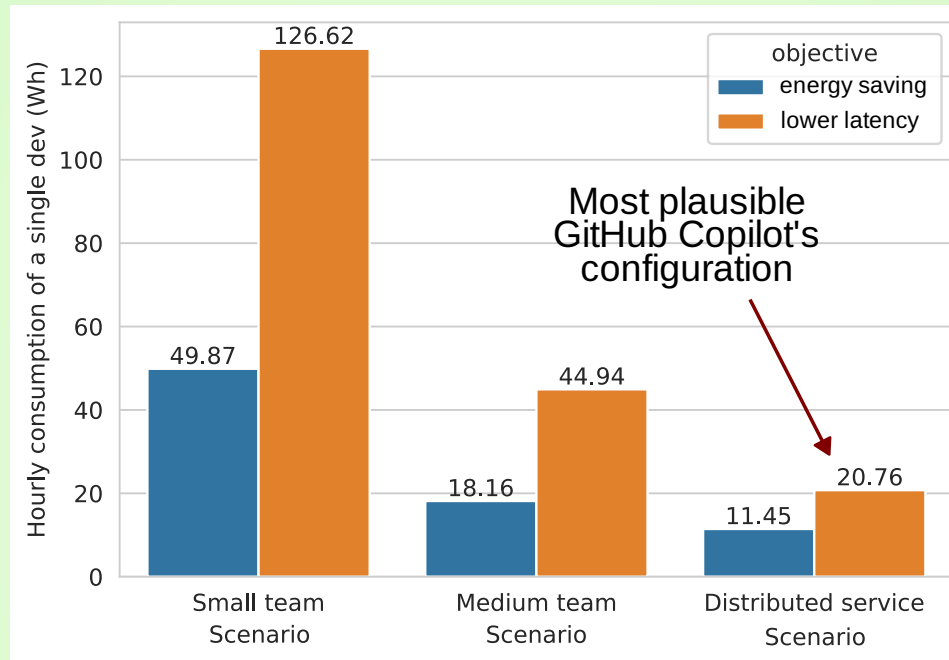


Fig. III.C. Energy consumption of a developer depending on the number of users on the inference server and the objective of the provider.

Using a code assistant like GitHub Copilot
could cost up to **20W**.

Takeaways

- *AssistantTraces* could enable further research into code assistants.
- Because of automatic generations, code assistants **waste a lot of ressources.**
- Code assistants have an **important impact on the energy consumption** of a developer.

[2] Coignon, T., Quinton, C., & Rouvoy, R. (2024). Green My LLM: Studying the key factors affecting the energy consumption of code assistants. Journal of Systems and Software (under minor revision).

Plan

1. Efficiency of LLM-generated code
2. Energy consumption of a code assistant
3. **Optimizing software with LLMs**
4. Conclusion

Introduction

**ECCO: Can We Improve Model-Generated Code Efficiency
Without Sacrificing Functional Correctness?**

Introduction

Learn to Code Sustainably: An Empirical Study on Green Code Generation

LEARNING PERFORMANCE-IMPROVING CODE EDITS

ECCO: Can We Improve Model-Generated Code Efficiency Without Sacrificing Functional Correctness?

LEARNING TO IMPROVE CODE EFFICIENCY

**Large Language Models for Energy-Efficient Code:
Emerging Results and Future Directions**

Introduction

Learn to Code Sustainably: An Empirical Study on Green Code Generation

LLM4EFFI: Leveraging Large Language Models to Enhance Code Efficiency and Correctness

LEARNING PERFORMANCE-IMPROVING CODE EDITING

ECCO: Can We Improve Model-Generated Code Efficiency Without Sacrificing Functional Correctness?

LEARNING TO EDIT

Generating Energy-Efficient Code via Large-Language Models – VCY
Where are we now?

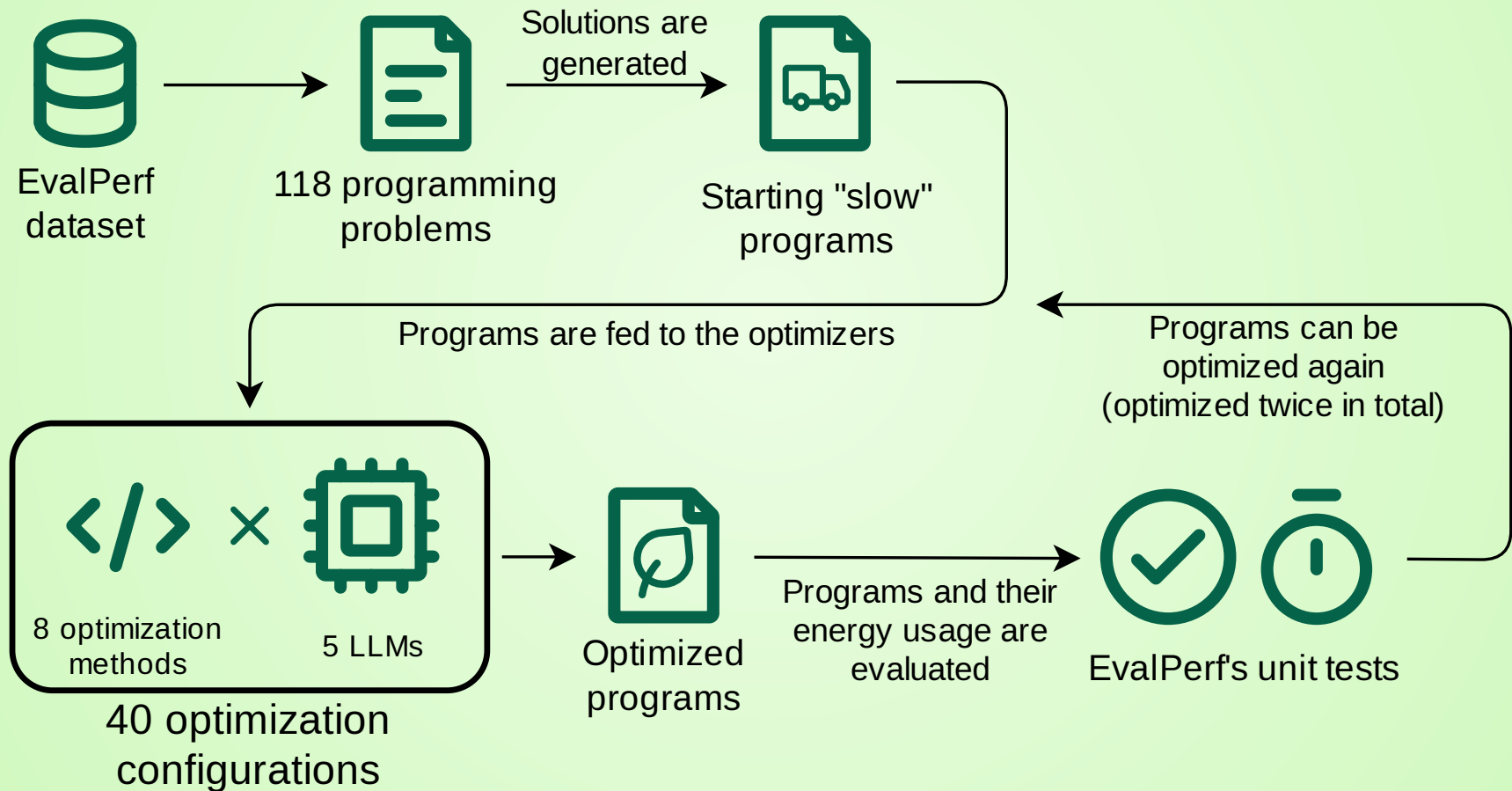
Large Language Models for Energy-Efficient Code: Emerging Results and Future Directions

Can We Make Code Green? Understanding Trade-Offs in LLMs vs. Human Code Optimizations
EFFI-LEARNER: Improving Efficiency of Generated Code via Self-Optimization

Research Question

*Is it energetically worth it to optimize code
using LLMs?*

Method



Method

Main Metrics

- **Energy reduction** → How much energy was reduced by the optimization. (e.g. EnergyReduction=0.6 → 40% less energy consumed)
- **Break-Even Point (BEP)** → Number of runs needed before an optimization is energetically profitable.

$$BEP = \frac{\text{Energy cost of optimization}}{\text{Energy saved per run}}$$

Results

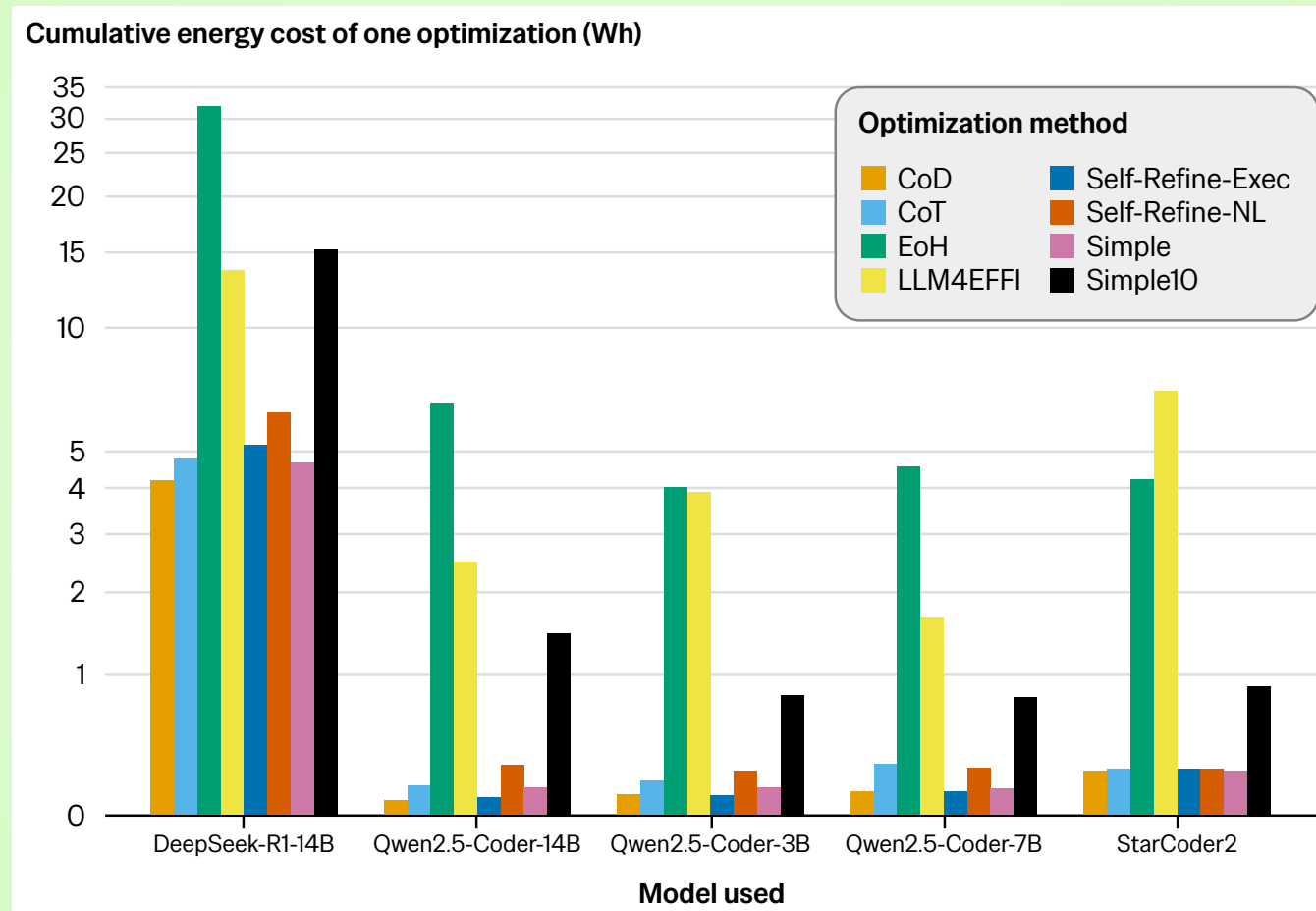


Fig. IV.A. Energy cost of an optimization depending on the model and method used.

Results

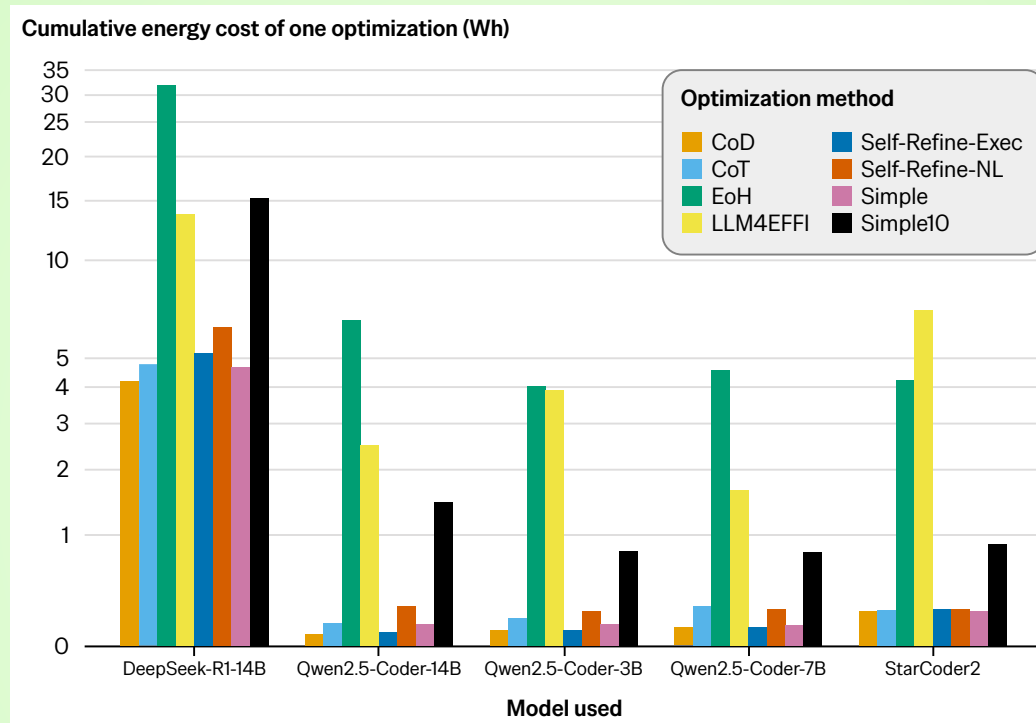


Fig. IV.A. Energy cost of an optimization depending on the model and method used.

Optimization costs are highly dependent on the model and method used.

Results

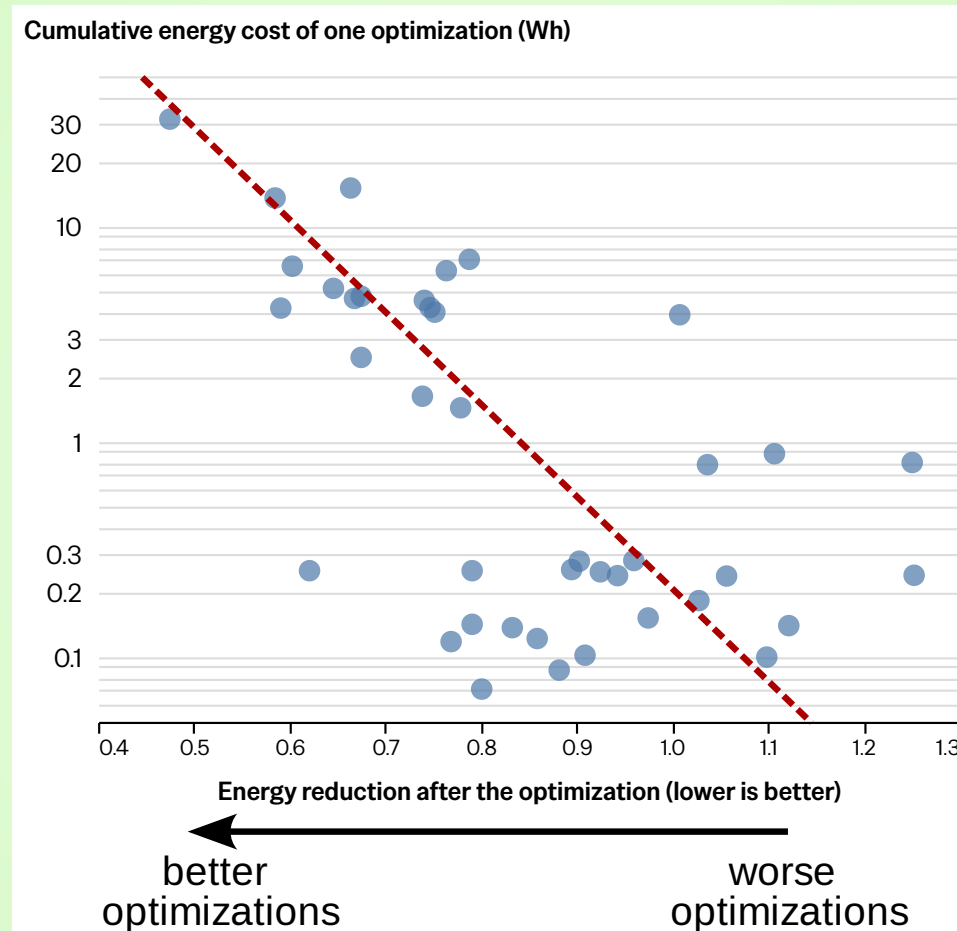


Fig. IV.B. Energy consumption of the optimization and its energy savings.
Each point is an optimization configuration.

Results

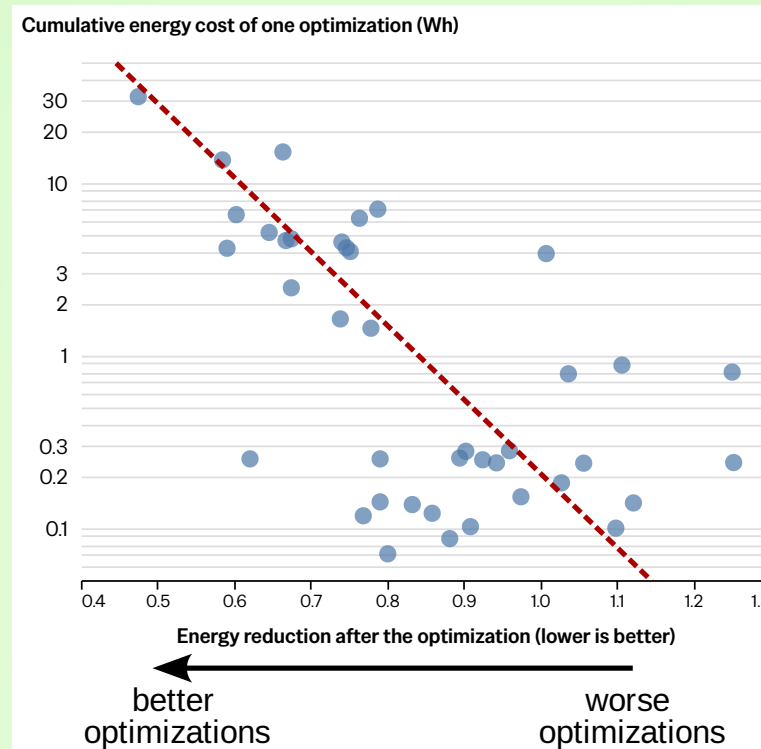


Fig. IV.B. Energy consumption of the optimization and its energy savings.
Each point is an optimization configuration.

Configurations that produce better
optimization also consume **more energy**.

Results

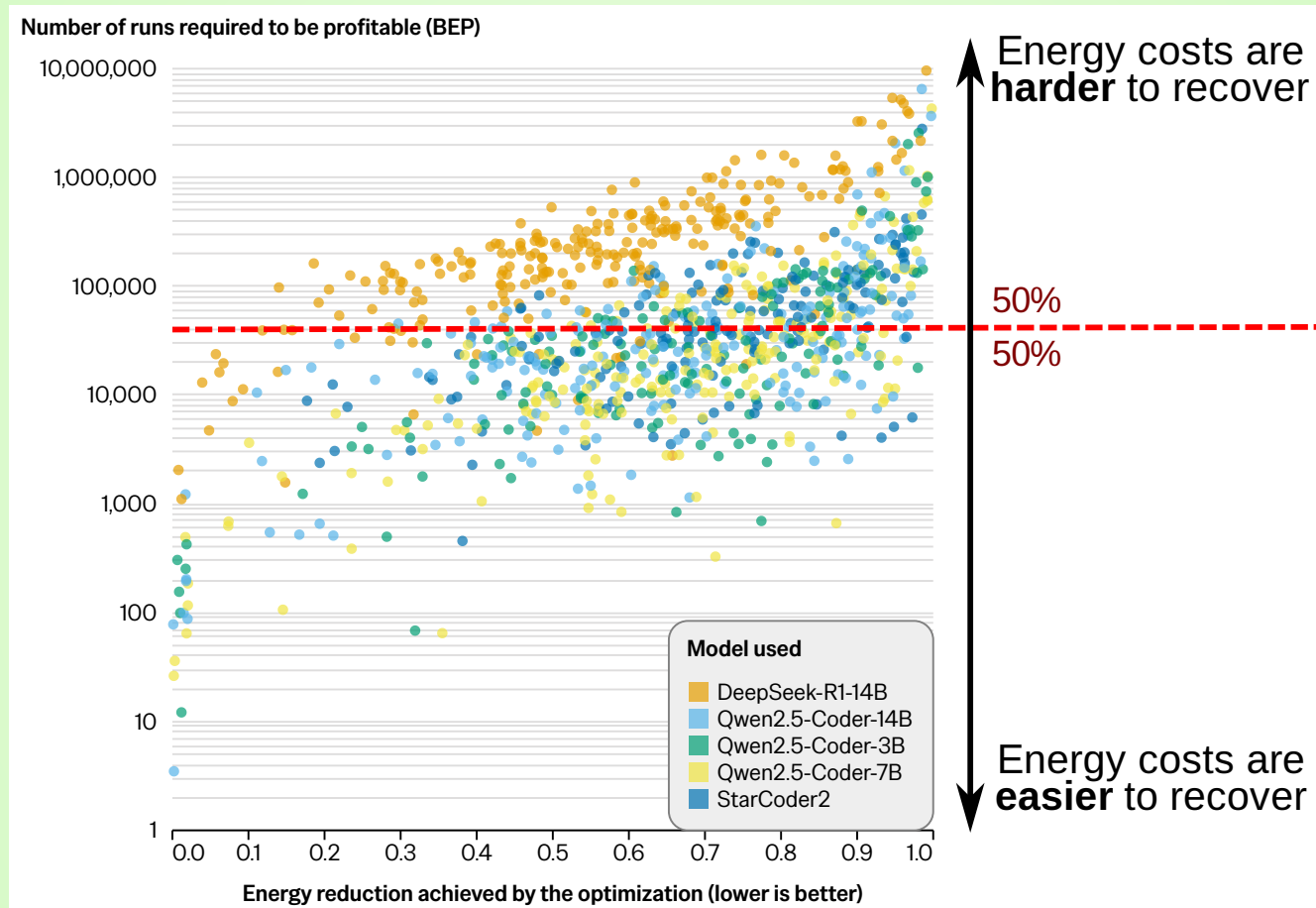


Fig. IV.C. Energy recoverability of the optimizations, and their energy reduction. Each point is one optimization. A sample of 1000 optimizations is shown.

Results

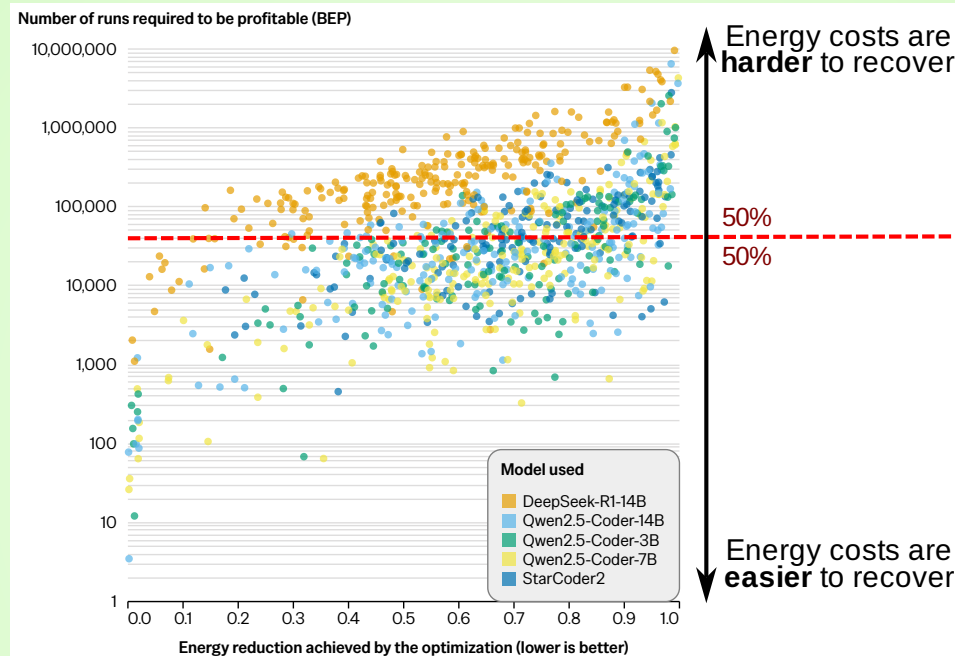


Fig. IV.C. Energy recoverability of the optimizations, and their energy reduction. Each point is one optimization. A sample of 1000 optimizations is shown.

Most optimizations require **thousands of executions** of the optimized program to be profitable.

Takeaways

- LLM-based code optimization **can effectively reduce software's energy consumption.**
- But, **we should be careful** about when and how we use LLM-based code optimization.

[3] Coignon, T., Quinton C., Rouvoy R., When Faster Isn't Greener: The Hidden Costs of LLM-Based Code Optimization. 40th IEEE/ACM International Conference on Automated Software Engineering, ASE 2025.

Plan

1. Efficiency of LLM-generated code
2. Energy consumption of a code assistant
3. Optimizing software with LLMs
4. **Conclusion**

In summary - What we did

- We studied the efficiency of LLM-generated code.
- We built a dataset of development traces and measured the energy consumption of code assistants like GitHub Copilot.
- Finally, we evaluated the energy profitability of LLM-based optimization.

In summary - What we found

- LLMs produce code with comparable efficiency that is on average slightly more efficient than human-written code, irrespective of the adopted model.
- Code assistants' energy consumption is quite high and could be easily reduced by changing usage patterns.
- LLM-based optimization can make software more efficient but should be carefully used as it can easily lead to energy waste.

Digression

The rebound effects of AI

Digression

The rebound effects of AI

*[...] improvements in efficiency often lead to **cost reductions** that provide the possibility to **buy more** of the improved product.*

Source: *Rebound effects of price differences*, Thiesen et al.

Digression

The rebound effects of AI

- **Material rebound effects:** More hardware is being produced.
- **Economic rebound effects:** AI is getting cheaper, and gets integrated everywhere.
- **Behavioral rebound effects:** We get used to AI services and use them even when not necessary.

Source: *From Efficiency Gains to Rebound Effects: The Problem of Jevons' Paradox in AI's Polarized Environmental Debate*, Luccioni, Strubell & Crawford.

Digression

The rebound effects of LLMs

- More and more AI-tools and users.
- Easier to optimize code → Wasteful optimizations.
- Faster development → More software is produced.

Efficiency is not a substitute for sobriety

Future Works

Short-term perspectives

- Improving the evaluation of code assistants.
- Wider configuration and hardware studies on code assistants.
- Broader evaluation of LLM-based code optimization.

Future Works

Long-term perspectives

- Standardized sustainable code benchmark and metrics.
- Designing an energy-aware code assistant.
- End-to-end life cycle assessment of code LLMs.

What I am going to do next

- Ongoing
 - Boavizta - **BoAmps project**.
 - Improving the energy efficiency of LLM deployment (w/ I. Malavolta, V. Stoico & P. Lago).
- Upcoming
 - Modeling the environmental impact of AI deployments in datacenters (w/ A. Ligozat, A. Bugeau & G. Guennebaud).

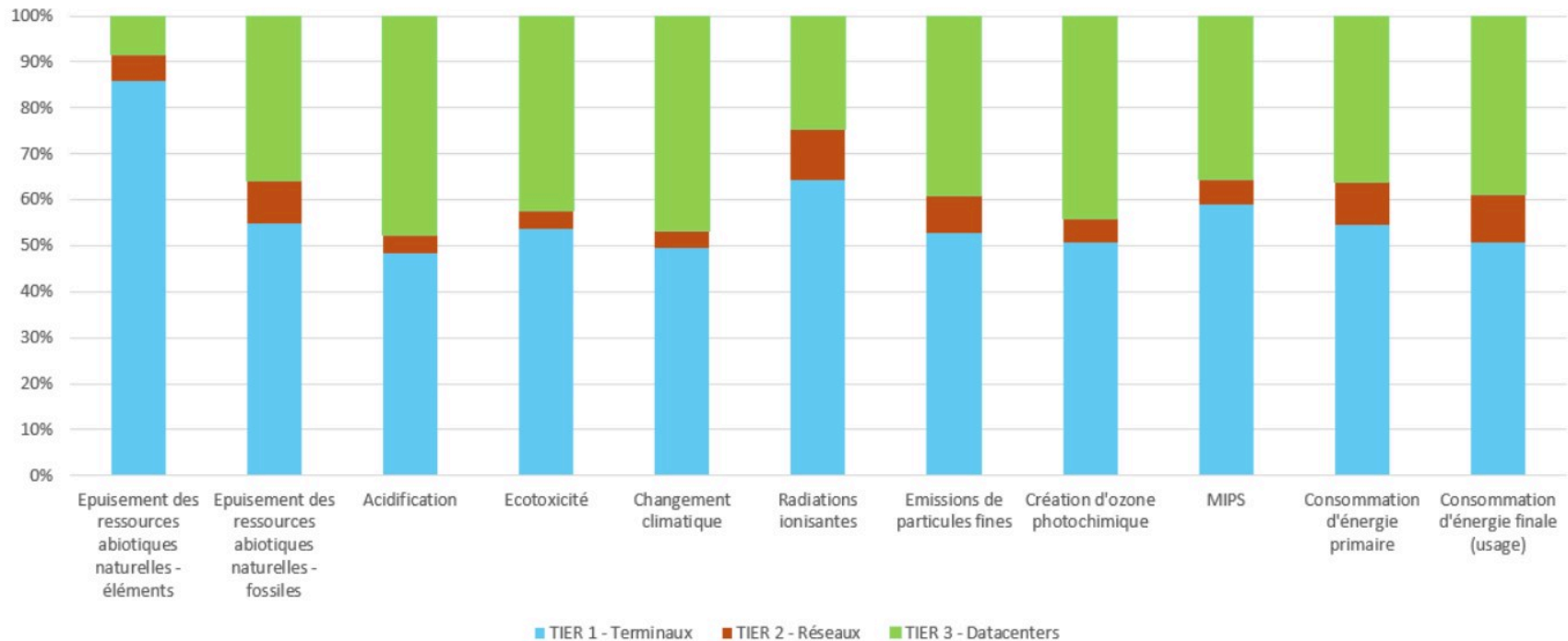
Closing words

Thank you for listening!

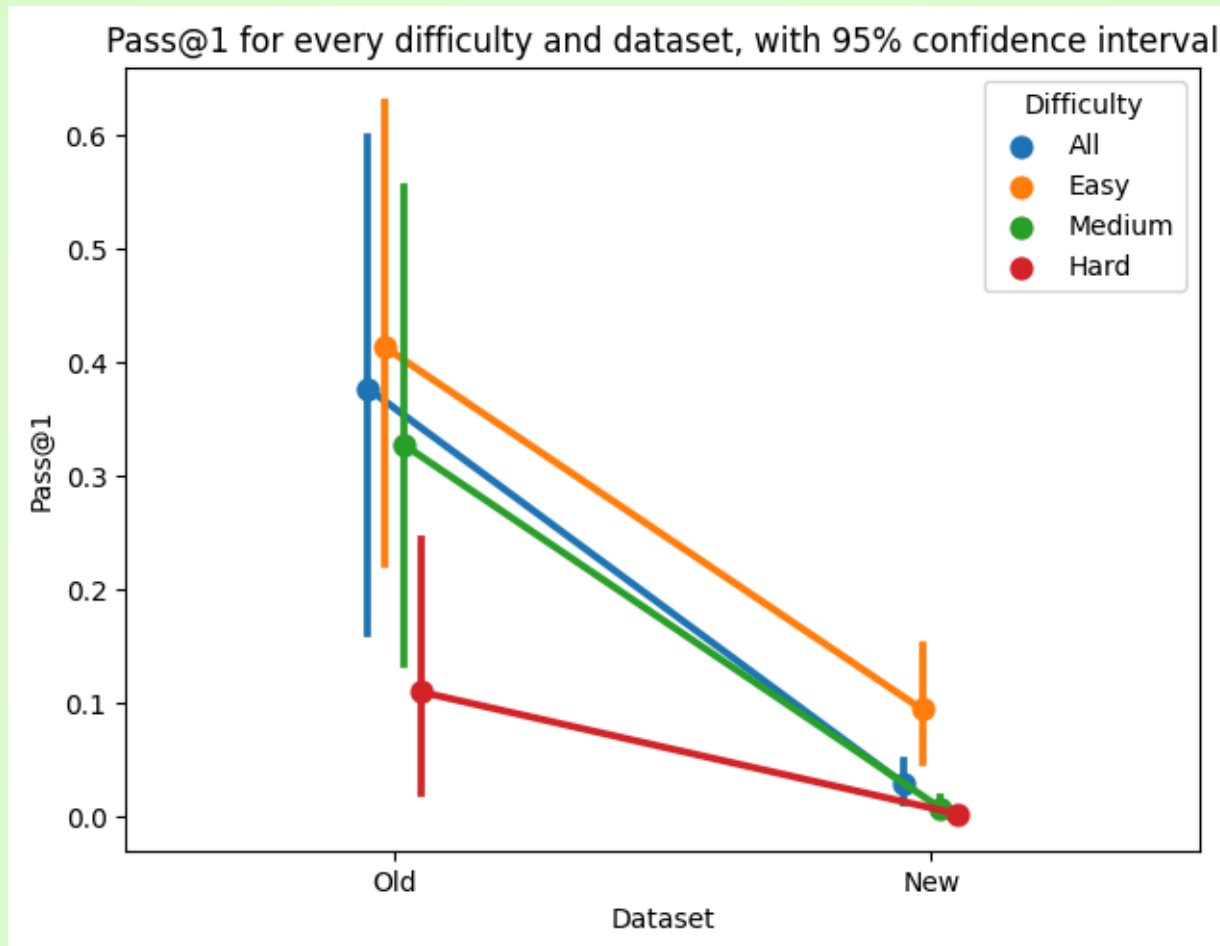
Backup slides



Répartition des impacts selon les 3 tiers

*Figure 2 – Décomposition des impacts par tier des équipements et infrastructures numériques*

Source: ADEME



LLMs are strongly data-contaminated by Leetcode questions.

LLMs produced correct code 10 times more often on older questions than on newer questions.

→ Leetcode is not a very reliable benchmark

Backup slides - Green My LLM

Various factors strongly impact the energy consumption and latency of code assistants.

Increasing the number of concurrent developers improves energy efficiency, but risks server saturation.

Streaming requests and manually triggering generations both strongly reduce energy consumption and latency. Quantization methods and the choice of LLM also affect latency, with smaller models showing better efficiency. Adjusting the number of GPUs is crucial for optimizing energy use. Our findings, therefore, confirm the importance of carefully selecting and optimizing these factors.

Scenario	Small team (dedicated server)		Medium team (dedicated server)		Distributed service	
	frugal	performance	frugal	performance	frugal	performance
Number of concurrent developers	5	5	20	20	75	50
Model	StarCoder2-7B	StarCoder	StarCoder2-7B	StarCoder	StarCoder2-7B	StarCoder
Quantization method	-	-	-	-	EETQ	-
Number of GPUs	1	4	1	4	4	4
Streaming	✗	✓	✗	✓	✗	✓
Manual trigger emulation	✓	✗	✓	✗	✓	✗
Average latency (s)	6.4	1.2	7.7	1.7	16.5	2.3
Average server power (W)	249.3	633.1	363.2	898.8	858.9	1038.2
Energy per 1000 generation requests (Wh)	12.0	30.7	0.9	2.2	0.1	0.4
Energy per hour per developer (Wh)	49.9	126.6	18.2	44.9	11.4	20.8
CO2 emissions per hour per developer (g)	2.8	7.1	0.9	1.0	0.6	1.2

Table 2: Optimal configurations for each scenario and objective, and their energy and latency impacts.

Metric	Independent variable	Groups	Test used	Stat	P-value	Significant
Ratio of accepted suggestion over rejected suggestions	Computer experience	professional, student	t-test	-2.496	0.022	✓
	GitHub Copilot familiarity	regularly, sometimes, never used	ANOVA	1.743	0.205	✗
	Coded Java in the last year	did code in java, didn't code in java	t-test	-0.570	0.576	✗
	Finished	finished, didn't finish	t-test	0.127	0.900	✗
Requests per minute	Computer experience	professional, student	t-test	0.808	0.429	✗
	GitHub Copilot familiarity	regularly, sometimes, never used	ANOVA	1.034	0.377	✗
	Coded Java in the last year	did code in java, didn't code in java	t-test	0.243	0.811	✗
	Finished	finished, didn't finish	t-test	0.799	0.434	✗

Table 1: Inferential statistics of the effect of three independent variables on two metrics. The significance was asserted using a Benjamini-Hochberg correction with a False Discovery Rate of 0.20

Results

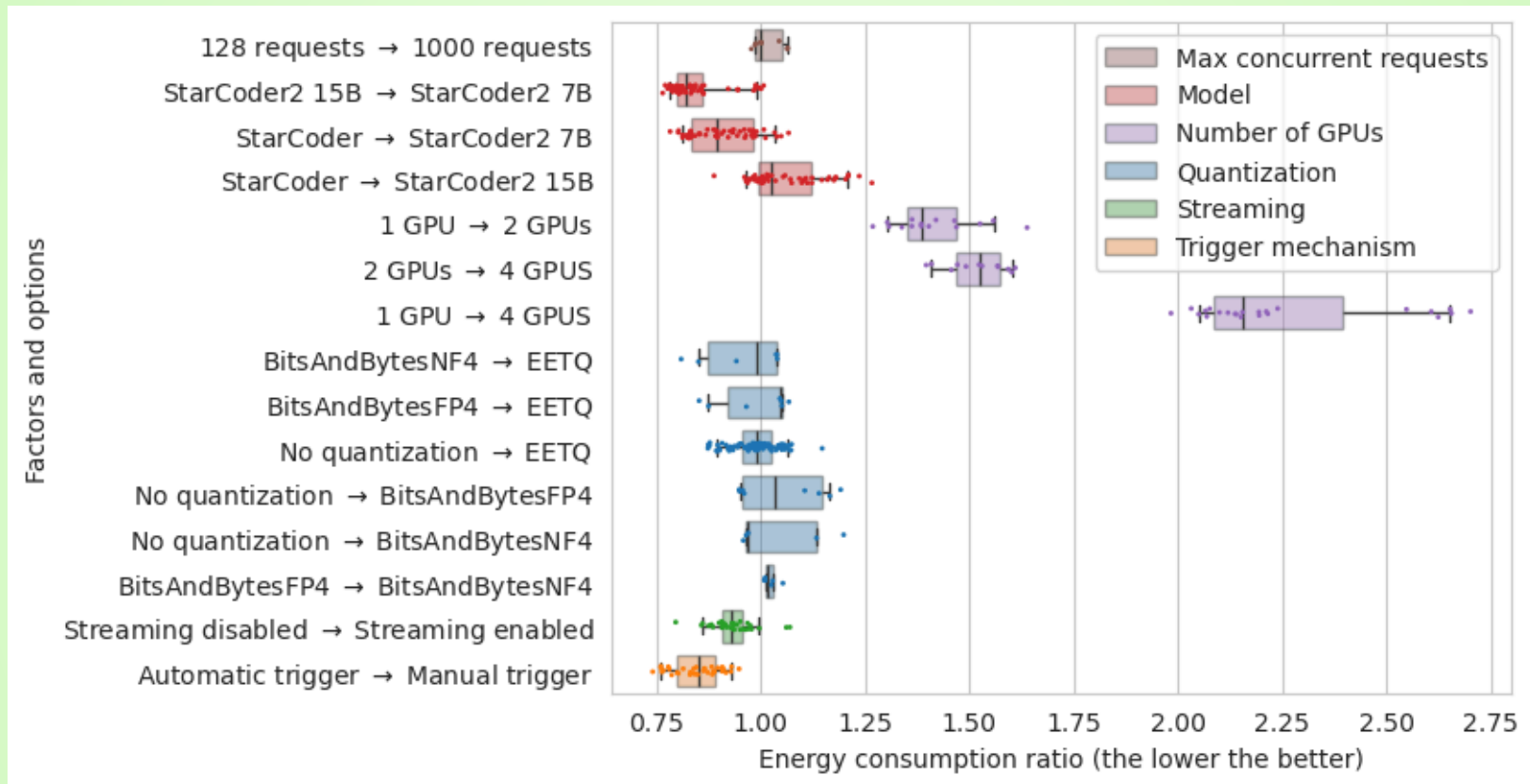


Fig III.A. Energy impact ratio from switching from one option to another. Points correspond to the ratio in energy when comparing neighboring configurations.

Backup slides - LLM-based code optimization

Method

EvalPerf

- Contains 118 programming problems
- Each problem has multiple possible solutions that have varying efficiency
- Contains both correctness-assessing tests and **performance-assessing** tests.
 - Allows us to better discriminate programs with slightly varying efficiency