Project Report

Sustainable Software

Group 9

TU Delft

Project Report

Sustainable Software

by

Junzhe Li	6200133
Yiming Chen	5541786
Mustafa Yücesan	5655544

Date: October 23, 2025
Course: Designing Sustainable ICT Systems
Responsible Lecturer: Przemysław Pawełczak

TU Delft

Contents

1 Experiment Design and JIT Optimization 1
1.1 Experiment Design (Food Ordering App Use Case) . . . 1
1.1.1 Goal and Scenario . . . 1
1.1.2 Assumptions . . . 1
1.1.3 Reproducibility of Experiments . . . 2
1.1.4 Daily Load Plan . . . 2
1.2 JIT Optimization . . . 2
1.2.1 JIT: what it is and why it matters . . . 2

2 Environmental Impact Assessment 3
2.1 Definition of Sustainability . . . 3
2.2 System Boundary . . . 3
2.3 Carbon Footprint Estimation . . . 4
2.3.1 Power Usage Efficiency and Grid Carbon Intensity . . . 4
2.4 Sensitivity Analysis . . . 4
2.4.1 Parameter 1: Carbon Intensity of Electricity . . . 5
2.4.2 Parameter 2: Server Power Consumption Profile . . . 5

3 Reflection 6
3.1 Reflection on the Design Process . . . 6
3.2 Self-Reflection on Energy Efficiency Results . . . 6

A Supporting Material 8
A.1 Experimental Setup . . . 8
A.1.1 Flags under test . . . 8
A.2 Plots . . . 8
A.3 Blended Carbon Intensity Calculation . . . 8
A.4 Energy Calculation Formula . . . 11
A.5 Sensitivity Analysis In-Depth . . . 11
A.5.1 Parameter 1: Carbon Intensity of Electricity . . . 11
A.5.2 Parameter 2: Server Power Consumption Profile . . . 12

Bibliography 13

1

Experiment Design and JIT Optimization

1.1. Experiment Design (Food Ordering App Use Case)

1.1.1. Goal and Scenario

Food delivery companies release little internal data, making it difficult to validate experimental scenarios. Global data, such as DeliveryHero's report of 11 million orders in one day across 70 countries [6], is unreliable for specific markets due to significant variations in ordering behaviour and population size. Local markets exhibit unique patterns; for instance, Dutch services typically start at 16:00, with weekends being the busiest. Publicly available datasets are often unsuitable. A Kaggle dataset on the topic [9] is too limited, containing only 11,399 records from a single Indian company and focusing narrowly on delivery speed. The use of "hyper-local" analytics by companies like JustEat [7] confirms that consumer behaviour varies significantly by area. Due to the lack of a suitable public dataset, assumptions are necessary to create a simplified benchmark for analysis, the validity of which will be discussed in Section 2.4.

1.1.2. Assumptions

Due to the lack of public data, several assumptions will have to be made.

A. It is assumed that with the 20 countries JustEat operates in, while local behaviour is different, overall they cancel each other out, resulting in a uniform distribution.
B. The assumption is that if Just Eat were as big as Delivery Hero, it would have a similar success.
C. On anecdotal evidence, it is assumed that most order activity is after 16:00, while before 16:00 there is hardly any order activity at all as most restaurants don't allow it before that time.
D. On anecdotal evidence, the assumption is that hotspot order activity is on Fridays, Saturdays and Sundays.
E. While behaviour will be different over the whole year (e.g. less order activity in summer), it is assumed it balances out over a whole year.
F. It is assumed that most of the workload is caused by placing orders; and other factors are negligible further defined in the system boundary.
G. It is assumed orders take 10% of all activity so average concurrency is 420 during each stage.
H. It is assumed the energy consumption within a 24 hour span is identical to the rest of the year.

Based on the above assumptions, a simplified model is created:

DeliveryHero achieved a record 11 million orders in one day (127 orders/second). Since JustEat is about a third of their size, its estimated potential for a record day is approximately 3.67 million orders, averaging 42 orders per second. Order flow is expected to be highest after 4 PM on peak days like Friday, Saturday, and Sunday.

Goal. Quantify how different JIT (Just-In-Time) configurations affect *latency, throughput, and energy* on an application-style server under a realistic, time-varying load.

Scenario. We simulate a full day as a varying workload: quiet overnight and early morning (CPU-only), ramp-

ing through late morning toward lunch (introducing some FILE work), easing in the afternoon, peaking again at dinner (more FILE), then tapering late evening.

1.1.3. Reproducibility of Experiments

The tests and optimizations will be conducted on the experimental setup defined in Appendix A.1. A known challenge is that different hardware configurations can yield significantly different performance results. Even on identical hardware, minor variations between test runs are expected. However, this is not a primary concern for this study, as its purpose is not to establish a hardware benchmark.

Rather, the intent is to measure the relative effect of specific JIT optimizations against a consistent baseline. The focus is on the percentage of improvement gained from an optimization, not the absolute performance metrics. The underlying hypothesis is that an optimization demonstrating a significant relative improvement on the experimental setup is likely to provide a similar directional improvement on other systems, even if the absolute performance values and the precise magnitude of the gain will differ.

1.1.4. Daily Load Plan

We split a day into easy time blocks with a target request mix and load as shown in Table 1.1

Phase	Time	Per-user RPS($r_u(t)$)	Concurrency	Mix (CPU/FILE)
Morning	06:00–10:30	0.2–1	2	1.0/0.0
Lunch	10:30–13:30	1–3	6	0.75/0.25
Afternoon	13:30–17:30	0.5–2	4	0.90/0.10
Dinner	17:30–20:30	1–3	8	0.65/0.35
Late	20:30–24:00	0.2–4	4	0.90/0.10
Night	00:00–06:00	0–1	2	1.0/0.0

Table 1.1: Profile of each stage of the simulated day

We model the app server of a food-ordering system where most traffic consists of small JSON `GET`/`POST` calls (browse menu, add to cart, login/status, checkout) plus occasional file transfers (images/icons). To study how **JIT configurations** behave under changing pressure, we assume that at peaks there are *more users* and they *act faster*. Each phase therefore specifies a *concurrency* (active users, uniformly downscaled to ~1/100 of a rough real-world estimate) and a phase-ramping *per-user request rate* $r_u(t)$. The open-loop driver launches

$$N(t) = \text{concurrency} \times r_u(t)$$

requests per second, sampled by a phase mix of two fixed request classes: **CPU** (`/work/cpu`, representing calculations) to represent lightweight JSON work, and **FILE** (`/work/files`, representing file transfer) with phase-specific file count/size to represent image/file transfers. The CPU/FILE share varies by phase.

1.2. JIT Optimization

We evaluate multiple JIT configurations over a *compressed* full day: one minute of wall time represents one hour of traffic, so a single **24-minute** run covers the entire daily profile. This keeps runs practical while preserving lunch/dinner peaks and off-peak segments. For each configuration we use the same day design, random seed, and scale factor (users and per-user rate), and we run the monitor at 1 Hz to collect process CPU/RSS, power, and cumulative energy.

1.2.1. JIT: what it is and why it matters

A Java JIT compiler translates frequently executed bytecodes into native code while the program runs. HotSpot uses tiered compilation: methods start in the interpreter (level 0), then are compiled by a fast, lightweight compiler (C1, levels 1–3) and, if deemed hot enough, recompiled by the optimizing compiler (C2, level 4). This design trades *warm-up time* (time spent interpreting/compiling) against *steady-state speed* (quality of generated code). Our day simulation stresses both: short off-peak phases emphasize warm-up behavior; lunch/dinner peaks emphasize steady-state throughput and tail latency.

2

Environmental Impact Assessment

This chapter provides a framework for assessing the project's environmental impact. To address the ambiguity of the term, a dedicated section will define the scope and meaning of "environmental impact" for this assignment.

2.1. Definition of Sustainability

In software development, sustainable software is designed to function effectively while minimising its carbon footprint. This project aims to simulate and optimise the workload of an existing system rather than building a new sustainable one from scratch.

As established in Section 1, this project aims to iteratively improve a simulated workload based on sustainability principles. The environmental impact is assessed within the context of a Food Ordering Application, for which a corresponding workload has been defined. This chapter uses the JIT compilation results from Section 1 to measure this impact.

2.2. System Boundary

To manage the resource-intensive nature of an impact assessment on an expansive food ordering system, a focused system boundary is defined, formalizing the hotspots introduced in Section 1. The core of the system is a back-end Spring Boot API server, which handles the primary workload and is optimized through a Just-In-Time (JIT) compiler, making processing server-centric. This renders client-side hardware specifications inconsequential to the analysis. As outlined in Table 2.2, the assessment's scope is strictly confined to cloud infrastructure, data transfer, and backend services, though it also considers relevant third-party digital services. A one-year analysis period is selected to establish a stable baseline, effectively neutralizing short-term anomalies and capturing the full spectrum of usage patterns, including daily, weekly, and seasonal variations. This timeframe is validated by the availability of annual data from comparable intermediary platforms like Thuisbezorgd or DeliveryHero. Consequently, several factors are explicitly out of scope: user device energy consumption, external services, food production, and packaging. Logistics services are also excluded due to

Principle	**Description**
Carbon Efficiency (CE)	Software emits the least amount of carbon possible.
Energy Efficiency (EE)	Software uses the least amount of energy possible.
Carbon Awareness (CA)	Do more when the electricity is cleaner and do less when it is dirtier.
Hardware Efficiency (HE)	Use the least amount of embodied carbon possible.
Measurement (M)	What you can't measure, you can't improve.
Climate Commitments (CC)	Understand the exact mechanism of carbon reduction.

Table 2.1: The basic principles of sustainable software engineering according to the Green Software Foundation.

Component / Life Cycle Stage
Included in Assessment (Digital Core)
Backend Services (Cloud servers, databases, APIs)
Associated Infrastructure (Data storage, networking, cooling)
Third-Party Digital Services (e.g., payment gateways, map APIs)
Excluded from Assessment (Physical World & User Devices)
Restaurant Operations (Food preparation, cooking energy)
Delivery Logistics (Fuel consumption, vehicle emissions)
Food Production & Packaging (Agricultural supply chain, containers)
End-User Devices (Electricity used by customer's phone/computer)
Hardware Lifecycle (Server Hardware)

Table 2.2: System Boundary for the Food Ordering Application Assessment

their minimal potential for JIT optimization, which would result in a negligible impact on the assessment.

2.3. Carbon Footprint Estimation

The primary focus is on the application's carbon footprint, which is determined by its Energy Efficiency (EE) and Carbon Efficiency (CE). EE ensures the software uses the least amount of energy possible, while CE ensures it emits the least amount of carbon possible from that energy use. As defined in Section 2.2, this analysis will assess the total carbon footprint over a one-year timespan. For this, the accumulated energy consumption will be used, and multiplied by a standard carbon intensity factor.

2.3.1. Power Usage Efficiency and Grid Carbon Intensity

The energy measurement is fully on the side on our backend, but in reality JustEat uses external services that use energy on their platform. To account for this a power usage efficiency factor will be used. JustEat used Amazon AWS a20 [4] in the past, which has a global PUE factor of 1.15 Amazon [8]. However, since recently they have moved to Google Cloud a20 [2], which has a general PUE factor of 1.09, which is much better a20 [5]. There will be a difference in carbon footprint depending on the electricity grid used. JustEat is active in around 20 countries, and the grid behaviour might vary widely within the countries itself. Because of this a weighted average is made based on a few countries using European data of 2021 a20 [1]. This single factor of 329 gCO2e/kWh is meant to represent the blended carbon intensity for the European countries represented. Please refer to Appendix A.3 to see how this factor is calculated.

Table 2.3 presents the energy usage per timeframe, calculated using the formula detailed in Appendix A.4. The yearly energy usage is based on Assumption H.

Table 2.3: Energy per time phase (J) over 24 minutes and totals; yearly energy is scaled from 24 minutes to 1 year

Config	Night	Morning	Lunch	Afternoon	Dinner	Late	Total (J)	Total (Wh)	Year (kWh)
Baseline	203.42	66.87	149.12	104.93	200.85	226.32	980.00	0.272	5.962
Interpreter-only	200.07	197.32	219.06	582.09	338.08	316.70	2194.77	0.610	13.352
C1-only	50.03	21.72	118.93	77.84	183.06	205.76	675.10	0.188	4.107
C2-only	213.74	216.98	231.82	131.05	204.42	225.35	1307.09	0.363	7.951
Compile sooner	22.78	15.55	109.50	54.89	177.81	171.66	563.44	0.157	3.428
Stable heap	23.25	16.25	109.41	52.11	178.99	169.29	559.81	0.156	3.406

2.4. Sensitivity Analysis

The environmental impact assessment presented thus far relies on a simplified model built upon several key assumptions, as outlined in Section 1.1.2. These assumptions, necessitated by the lack of public data, introduce inherent uncertainty into the final calculations. This section aims to identify the most significant sources of this uncertainty and quantify their potential effect through a sensitivity analysis. This analysis will

focus on two critical parameters whose variability can dramatically alter the outcome of the assessment: the **carbon intensity of electricity** and the **server power consumption profile**.

2.4.1. Parameter 1: Carbon Intensity of Electricity

The choice of this parameter is driven by the significant uncertainty introduced by **Assumption A**, which posits that the varied consumer behaviours and energy grids across the 20 countries JustEat operates in will "cancel each other out" into a uniform distribution. This is a substantial oversimplification. The carbon intensity of electricity (measured in gCO_2eq/kWh) varies dramatically between nations, depending on their energy mix (e.g., nuclear, coal, renewables). For instance, the electricity grid in France is far less carbon-intensive than the grid in Germany or Poland. Furthermore, **Assumption H**, which assumes an identical consumption profile year-round, ignores the temporal variations in grid carbon intensity. Therefore, *where* the computational workload occurs is as important as *how much* energy it consumes.

The calculated carbon footprint can vary from 110 kg to 7,000 $kgCO_2eq$—a factor of over 60—based solely on the geographic location of the computation. This shows the model's extreme sensitivity to Assumption A and highlights that an accurate environmental assessment is impossible without geo-located workload data. The details of the analsysis on this parameter can be found in Appendix A.5.1

2.4.2. Parameter 2: Server Power Consumption Profile

This parameter is chosen to challenge the implicit assumption that energy consumption is directly and linearly proportional to the workload (i.e., orders placed), as suggested by **Assumption F** and **Assumption H**. In reality, data centre hardware consumes a significant amount of power even when idle. Given that **Assumptions C and D** concentrate the workload into a narrow window (post-16:00 on weekends), the system's infrastructure remains idle for a large portion of the day. A model that ignores this constant idle power draw will severely underestimate total energy consumption.

The model created to demonstrate this estimates a total energy consumption of 1,240 kWh, which is more than double the 600 kWh calculated by the simplified model. This analysis reveals that ignoring the system's idle power—a direct consequence of the simplifying assumptions—leads to a gross underestimation of energy consumption by over 100%. The majority of the system's daily energy usage is dedicated to simply being operational, a critical factor not captured in the initial model. The details of the analsysis on this parameter can be found in Appendix A.5.2

3

Reflection

This chapter briefly reflects on our design choices and how they shaped the outcomes. Then discusses the results of our environmental impact assessment.

3.1. Reflection on the Design Process

We scoped the unit to the app server and modeled requests as small JSON with a compute surrogate and a file surrogate, as shown in Section 1.1.4. In hindsight the workload likely underestimates I/O. A real food-ordering stack leans on caches, databases, and the network with image delivery and occasional cold misses, while our execution was more CPU-bound. We also fixed per-request cost across phases and used an open-loop driver. These choices make attribution to JIT clear yet collapse real variance such as changing query plans, cache hit rates, and queueing. The user scaling factor is a pragmatic hardware-driven heuristic rather than an empirical mapping. A stronger mapping from laptop to server would improve realism, yet our choice preserves the workload shape and supports meaningful relative comparisons between JIT configurations.

Part 2 uses a simple impact model. We sample process power at 1 Hz, sum energy, and convert to CO_2e with a single PUE and a single grid-intensity factor. This keeps runs comparable but assumes constant power within each second and constant PUE and grid intensity. It also treats app-server energy as a proxy for the whole service and ignores embodied carbon. Sensitivity analysis shows how totals shift when the checkout fraction or the grid factor changes. With more time and hardware we would increase FILE and DB-like work, introduce a small distribution of latency and size, replay carbon-intensity traces, and validate on a second host.

In general, the design is strong for relative comparisons under a daily pattern and weak for absolute realism. What worked was clear attribution and reproducibility. What did not was representativeness and scaling.

3.2. Self-Reflection on Energy Efficiency Results

The experiment was straightforward, but the measured energy values were lower than expected due to limited resources and necessary assumptions. Nevertheless, the results were sufficient to compare different JIT configurations in relative terms. The main goal—to explore how much energy can be saved through JIT tuning—was achieved, as the percentage differences clearly highlighted the most efficient profiles. The simplified design ensured reproducibility. Overall, the experiment demonstrated that even small-scale simulations can yield meaningful insights into energy-saving potential when consistent conditions and relative comparisons are applied.

Config	Night	Morning	Lunch	Afternoon	Dinner	Late	Total
Baseline	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Interpreter-only	-1.65%	+195.0%	+46.9%	+454.9%	+68.3%	+39.9%	+124.0%
C1-only	-75.4%	-67.5%	-20.2%	-25.8%	-8.9%	-9.1%	-31.1%
C2-only	+5.1%	+224.6%	+55.5%	+24.9%	+1.8%	-0.4%	+33.4%
Compile sooner	**-88.8%**	**-76.7%**	**-26.6%**	-47.7%	**-11.5%**	-24.2%	-42.5%
Stable heap	-88.6%	-75.7%	**-26.6%**	**-50.3%**	-10.8%	**-25.2%**	**-42.8%**

Table 3.1: Relative energy difference (%) compared to Baseline per phase (negative = improvement / saving)

A

Supporting Material

A.1. Experimental Setup

- **Laptop:** ASUS TUF Gaming A15 (FA506QM), UEFI FA506QM.314 (2023-01-07).
- **OS / Kernel / DE:** Ubuntu 24.04.3 LTS, Linux 6.8.0-85-generic, GNOME 46.
- **CPU:** AMD Ryzen 7 5800H (Zen 3), 8 cores / 16 threads, L3 16 MiB (SMT enabled).
- **Memory:** 16 GiB RAM.
- **Battery:** ASUS A32-K55; reported full 59.7 Wh (design 90.2 Wh, ~66% health).

A.1.1. Flags under test

- **Baseline (default tiered JIT)**: HotSpot decides when to move methods from interpreter → C1 → C2. Expect good warm-up and strong peak performance.
- **Interpreter-only (`-Xint`)**: Disable all JIT. Every bytecode is interpreted. Expect very high CPU per request, very low throughput, large latency; may fail to keep up with open-loop load (we mark it as a control that can legitimately fail).
- **C2-only (`-XX:-TieredCompilation`)**: Skip C1; compile hot methods directly to C2. Expect longer warm-up (more time interpreting before C2 kicks in), the best steady-state throughput, and strong peak efficiency once hot.
- **C1-only (`-XX:+TieredCompilation -XX:TieredStopAtLevel=1`)**: Stop at level 1 (no C2). Expect very fast warm-up and low compile overhead, but lower peak throughput and slightly worse tail latency than Baseline/C2.
- **Compile sooner (`-XX:CompileThreshold=1000`)**: Methods reach C1/C2 with fewer invocations. Expect quicker escape from interpreter, earlier steady-state, but extra compile work up front; may raise early power slightly and reduce peak tails.
- **Stable heap (`-Xms1g -Xmx1g`)**: Not a JIT mode; fixes heap size to reduce GC noise.

A.2. Plots

A.3. Blended Carbon Intensity Calculation

The data, in gCO2e/kWh, includes Germany (348), Netherlands (339), Austria (82), Belgium (139), Bulgaria (398), Denmark (123), Ireland (332), Italy (234), Luxembourg (45), Poland (721), Slovakia (113), and Spain (165). Consequently, the grid carbon intensity factor is based solely on these European markets.

The estimated workload is based on the annual report of JustEat [3] and is meant as an informed **estimate** rather than a precise metric.

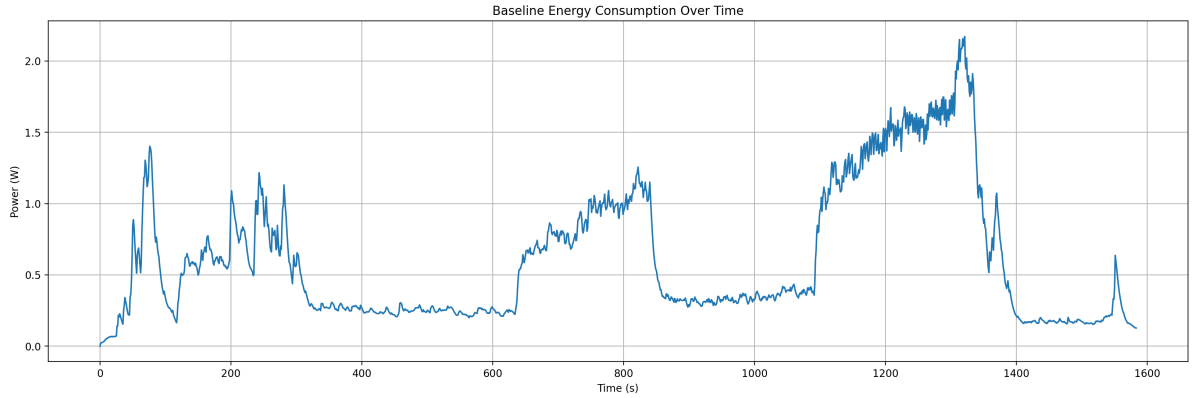

Figure A.1: Baseline (default tiered JIT): power over samples.

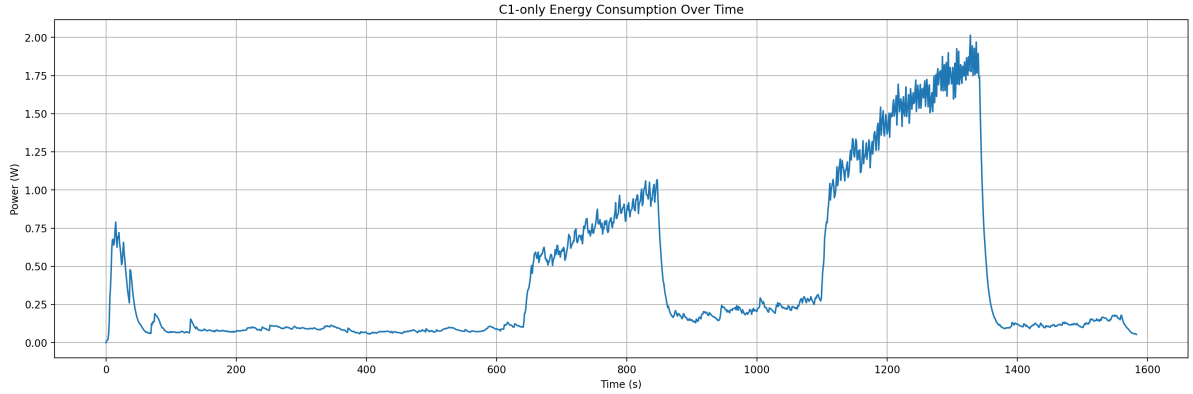

Figure A.2: C1-only: power over samples.

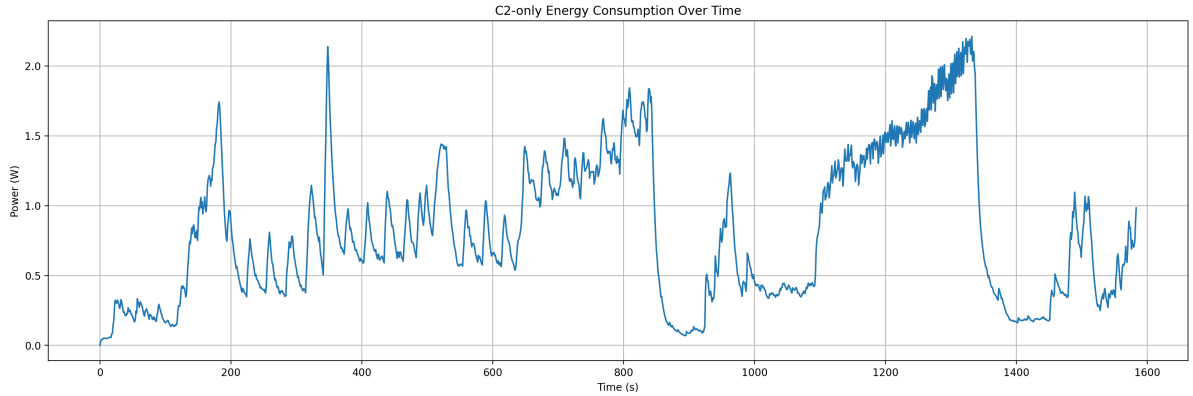

Figure A.3: C2-only: power over samples.

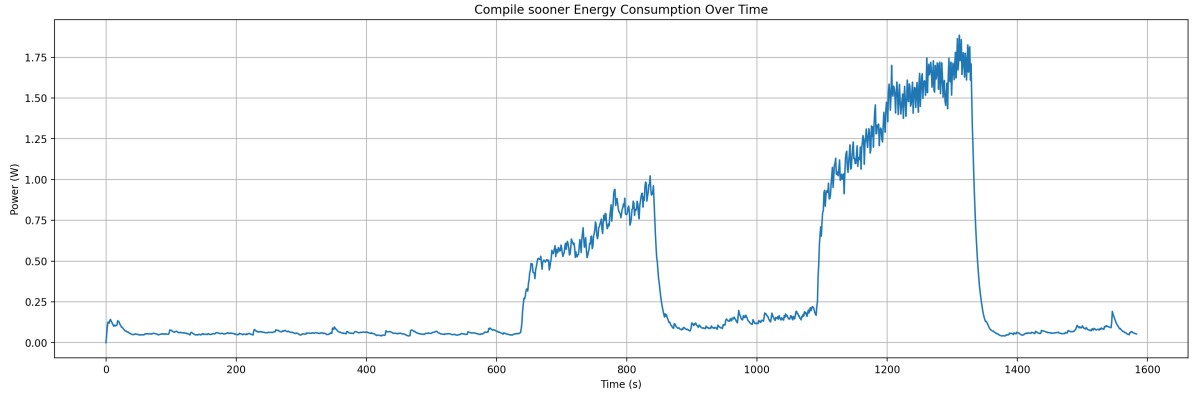

Figure A.4: CompileThreshold=1000: power over samples.

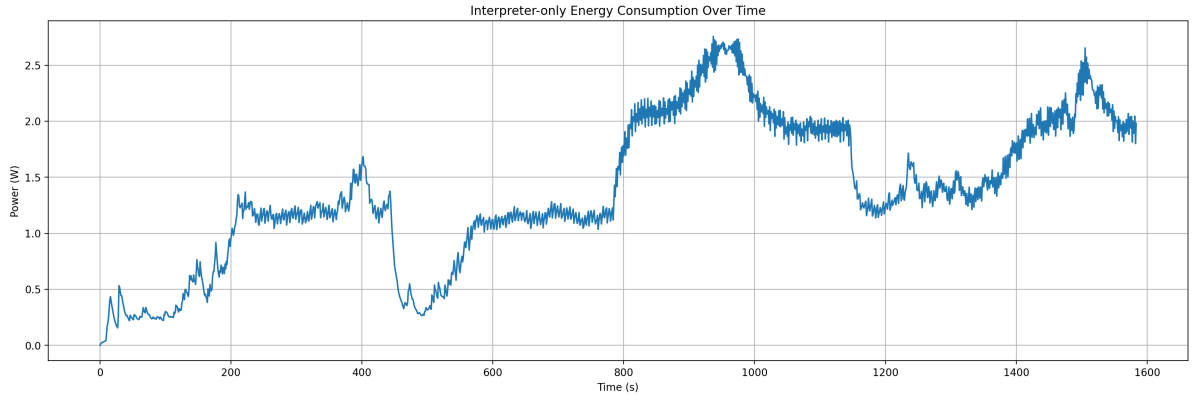

Figure A.5: Interpreter-only (-Xint): power over samples.

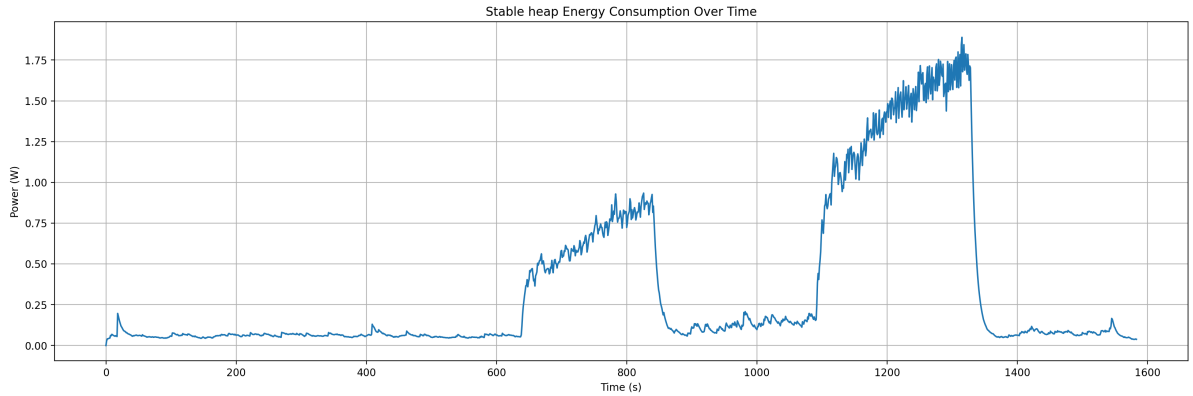

Figure A.6: Stable heap (-Xms1g -Xmx1g): power over samples.

Country	Estimated Workload	Grid Carbon Intensity (gCO2e/kWh)	Weighted Value
Germany	30%	348	104.4
Netherlands	25%	339	84.8
Poland	10%	721	72.1
Italy	8%	234	18.7
Spain	8%	165	13.2
Ireland	4%	332	13.3
Denmark	4%	123	4.9
Belgium	3%	139	4.2
Austria	3%	82	2.5
Bulgaria	2%	398	8.0
Slovakia	2%	113	2.3
Luxembourg	1%	45	0.5
Total / Weighted Average	**100%**		**328.9**

Table A.1: Calculation of the Weighted-Average Carbon Intensity Factor for Just-Eat Takeaway's specified European markets.

A.4. Energy Calculation Formula

This formula calculates the total grams of CO2 equivalent emissions (gCO_2e) from a given amount of energy in Joules, considering the Power Usage Effectiveness (PUE) and the grid's carbon intensity.

$$gCO_2e = \left(\frac{\text{Total Joules}}{3,600,000}\right) \times \text{PUE} \times \text{Grid Emission Factor}$$

Example Calculation

Here is a step-by-step example calculation.

1. **Convert Energy from Joules to kilowatt-hours (kWh):**

$$\text{Energy (kWh)} = \frac{500,000,000,000\ \text{J}}{3,600,000\ \text{J/kWh}} \approx 138,888.89\ \text{kWh}$$

2. **Apply the Power Usage Effectiveness (PUE):**

$$\text{Total Energy Consumption} = 138,888.89\ \text{kWh} \times 1.09 \approx 151,388.89\ \text{kWh}$$

3. **Apply the Grid Emission Factor to find total gCO_2e:**

$$\text{Total } gCO_2e = 151,388.89\ \text{kWh} \times 329 \frac{gCO_2e}{\text{kWh}} \approx 49,806,944.44\ gCO_2e$$

A.5. Sensitivity Analysis In-Depth

A.5.1. Parameter 1: Carbon Intensity of Electricity

A.5.1.1. Justification

This parameter is chosen to address the significant uncertainty introduced by **Assumption A**, which posits that the varied consumer behaviors and energy grids across the 20 countries JustEat operates in will "cancel each other out" into a uniform distribution. This is a substantial oversimplification, as the carbon intensity of electricity (measured in gCO_2eq/kWh) varies dramatically between nations depending on their energy mix (e.g., nuclear, coal, renewables). For instance, France's electricity grid is far less carbon-intensive than Germany's or Poland's. Furthermore, **Assumption H**, which assumes an identical consumption profile year-round, ignores temporal variations in grid carbon intensity. Therefore, *where* the computational workload occurs is as important as *how much* energy it consumes.

A.5.1.2. Sensitivity Analysis

To quantify this effect, we will calculate the total daily carbon emissions based on a hypothetical total energy consumption of 10,000 kWh, testing three scenarios for carbon intensity.

- **Baseline Scenario (European Average):** We assume the workload is evenly distributed, using an average carbon intensity for the European grid of **250 gCO_2eq/kWh**.

  ```
  Emissions = 10,000 kWh * 250 gCO2eq/kWh = 2,500,000 gCO2eq = 2,500 kgCO2eq
  ```

- **Best-Case Scenario (Low-Carbon Grid):** We assume the majority of the workload is processed in a region with a low-carbon grid, such as Sweden, with an intensity of **11 gCO_2eq/kWh**.

  ```
  Emissions = 10,000 kWh * 11 gCO2eq/kWh = 110,000 gCO2eq = 110 kgCO2eq
  ```

- **Worst-Case Scenario (High-Carbon Grid):** We assume the majority of the workload is processed in a region reliant on fossil fuels, such as Poland, with an intensity of **700 gCO_2eq/kWh**.

  ```
  Emissions = 10,000 kWh * 700 gCO2eq/kWh = 7,000,000 gCO2eq = 7,000 kgCO2eq
  ```

The results demonstrate that the calculated carbon footprint can vary from 110 kg to 7,000 kgCOeq, a factor of over 60, based solely on the geographic location of the computation. This finding reveals the model's extreme sensitivity to Assumption A and underscores that an accurate environmental assessment is impossible without geo-located workload data.

A.5.2. Parameter 2: Server Power Consumption Profile

A.5.2.1. Justification

This parameter challenges the implicit assumption, suggested by **Assumption F** and **Assumption H**, that energy consumption is directly and linearly proportional to the workload (i.e., orders placed). In reality, data center hardware consumes significant power even when idle. Because **Assumptions C and D** concentrate the workload into a narrow window (post-16:00 on weekends), the system's infrastructure is idle for most of the day. Consequently, any model that ignores this constant idle power draw will severely underestimate total energy consumption.

A.5.2.2. Sensitivity Analysis

To quantify this, we compare two models for calculating the total daily energy consumption of a server fleet. We assume the fleet has a maximum power draw of 100 kW and a constant idle power draw of 40 kW (40% of max). The workload is concentrated in an 8-hour peak window (16:00-24:00), during which the servers operate at an average of 75% capacity (75 kW).

- **Scenario A (Simplified Proportional Model):** This model, implied by the current assumptions, incorrectly links power draw only to active workload. It assumes 75 kW of power for the 8-hour peak period and neglects the 16 hours of idle time.

  ```
  Peak Energy = 75 kW * 8 hours = 600 kWh
  Idle Energy = 0 kW * 16 hours = 0 kWh
  Total Daily Energy = 600 kWh
  ```

- **Scenario B (Realistic Idle/Load Model):** This model accounts for the constant power draw of the infrastructure. The power draw is 75 kW during the peak and 40 kW during the 16-hour idle period.

  ```
  Peak Energy = 75 kW * 8 hours = 600 kWh
  Idle Energy = 40 kW * 16 hours = 640 kWh
  Total Daily Energy = 1,240 kWh
  ```

The realistic Idle/Load model estimates a total energy consumption of 1,240 kWh, which is more than double the 600 kWh calculated by the simplified model. This analysis reveals that ignoring the system's idle power, a direct result of simplifying assumptions, leads to a gross underestimation of energy consumption by over 100

Bibliography

[1] Greenhouse gas emission intensity of electricity generation in europe, 2015. URL https://www.eea.europa.eu/de/ims/greenhouse-gas-emission-intensity-of-1.

[2] Just eat takeaway.com's migration to google cloud, 2023. URL https://pages.xebia.com/just-eat-takeaway-google-cloud-xebia.

[3] Just eat takeaway.com - investors - results, reports presentations - annual reports, 2024. URL https://www.justeattakeaway.com/investors/results-reports-presentations/annual-reports/default.aspx.

[4] Aws innovator: Just eat takeaway | case studies, videos and customer stories, 10 2025. URL https://aws.amazon.com/solutions/case-studies/innovators/just-eat/.

[5] Effectiviteit van energieverbruik – google datacenters, 2025. URL https://datacenters.google/efficiency/.

[6] Delivery hero hits record of 11 million orders in a day, reinforcing global leadership in food delivery | delivery hero, 2025. URL https://www.deliveryhero.com/newsroom/delivery-hero-hits-record-of-11-million-orders-in-a-day-reinforcing-global-leadership-in-food-d

[7] Daniel Bos . Launching jet 360: The power of data analytics at just eat takeaway.com, 04 2023. URL https://newsroom.justeattakeaway.com/en-WW/225568-launching-jet-360-the-power-of-data-analytics-at-just-eat-takeaway-com/.

[8] Amazon. Aws cloud - amazon sustainability, 2023. URL https://sustainability.aboutamazon.com/products-services/aws-cloud.

[9] Gaurav Malik. Food delivery dataset, 2022. URL https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset/data.