TRAFFIC: Testbed foR Assessing energy eFFiciency In throughput Computing

Loïc Guibert
HES-SO
Fribourg, Switzerland
loic.guibert@hefr.ch

Sébastien Reynaud
HES-SO
Fribourg, Switzerland
sebastienreynaud@proton.me

Olivier Weppe
INSA Rennes
Rennes, France
olivier.weppe@insa-rennes.fr

Kristjon Ciko
University of Oslo
Oslo, Norway
kristjoc@ifi.uio.no

Michael Welzl
University of Oslo
Oslo, Norway
michawe@ifi.uio.no

Sébastien Rumley
HES-SO
Fribourg, Switzerland
sebastien.rumley@hefr.ch

Abstract

Information and Communication Technologies (ICT) represent a significant share of global resource usage. Notably, the energy consumption of data centres has emerged as a critical concern, escalating rapidly, especially with the rise of generative Artificial Intelligence (AI) models. This surge in energy demand calls for efforts focused on measuring and reducing the energy and carbon footprints of components used in ICT service delivery. However, the variety and multitude of devices involved in these services make it challenging to accurately measure and evaluate these footprints.

In this paper, we propose an alternative approach to assess energy impacts by considering data centres for what they are: *throughput computing systems offering ICT services*. We define a simple experimental testbed and evaluate several machines with different hardware capacities. These machines serve two ICT services: a loop incrementing a counter until a defined limit, and an AI inference predicting the next token based on an input.

Our experiment highlights three main take-aways: (a) the CPU usage is a *poor predictor* of the power consumption, (b) the energy or CO_2 quantity associated with a service visit *highly depends on the total load* the service is facing, and (c) modern machines *do not yield better energy figures* compared to older ones in all circumstances.

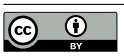
CCS Concepts

• **Hardware** → **Interconnect power issues**; *Impact on the environment*; • **Information systems** → Web services; • **Computing methodologies** → *Artificial intelligence*; • **General and reference** → **General conference proceedings**; **Measurement**; • **Social and professional topics** → Sustainability.

Keywords

Energy, Throughput computing, ICT carbon footprint

ACM Reference Format:
Loïc Guibert, Sébastien Reynaud, Olivier Weppe, Kristjon Ciko, Michael Welzl, and Sébastien Rumley. 2025. TRAFFIC: Testbed foR Assessing energy eFFiciency In throughput Computing. In *22nd ACM International Conference on Computing Frontiers (CF Companion '25), May 28–30, 2025, Cagliari, Italy.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3706594.3727961



CF Companion '25, Cagliari, Italy

ACM ISBN 979-8-4007-1393-4/25/05
https://doi.org/10.1145/3706594.3727961

1 Introduction

Information and Communication Technologies (ICT) already consume a significant share of the world's resources. With the advent of generative Artificial Intelligence (AI), the world is witnessing a sharp increase in the energy usage of data centres [2, 4]. In its recent report, the Lawrence Berkeley National Laboratory estimates that data centres could account for 12% of the total energy consumption in the US by 2028 [31]. These concerns have spurred numerous initiatives to improve the energy or carbon footprint of ICT.

This *green enthusiasm* is welcome, but sometimes lacks scientific rigour. For starters, if one is interested in improving a metric, it is of uttermost importance to be able to properly measure this metric first in an appropriate unit, using an appropriate measurement device. This implies that the measurement device has been validated and calibrated. If, when trying to improve things, one has the eye on the wrong gauge, or a malfunctioning gauge, the effects of optimisation may fall short of expectations.

Let us take the ecological design of websites as an example. There have been initiatives to promote lighter and sober websites [15]: tools such as *Ecograder* [26] or *Website Carbon Calculator* [33] provide estimates of the amount of CO_2 associated with one visit to a website. For instance, the website of the *HotCarbon* conference[1] dissipates 0.06 gCO_2 per visit according to *Website Carbon Calculator*. Efforts to promote sober designs are certainly worthwhile, but focusing on this sole gCO_2*/visit* metric would be foolish, as we will show in this paper.

Indeed, the model behind the calculator has not been (and cannot be) thoroughly validated. Validating it would typically require: (a) that the footprint of each and every device involved in *one arbitrary visit* is measured once with and without this visit, and (b) that a sophisticated *allocational* or *consequential* analysis [14] is applied to estimate the footprint associated with *one arbitrary visit*, since the system is not made for this *arbitrary visit* in particular. Given the complexity of the *cyberspace*, properly measuring the energy or CO_2 associated with *one arbitrary action* is nearly infeasible. Furthermore, a good knowledge of the context is required to realise an *allocational* or *consequential* analysis. To refer back to our example,

[1] https://hotcarbon.org/

one must in particular determine how many *other visits* the web server must handle, i.e., the load the service is facing.

In this paper, we propose to take a new approach to assess energy impacts of *visits of ICT services*. We propose to drastically simplify the *cyberspace* until a point where it becomes unequivocally measurable. We also propose to take simplifying assumptions about the context within which a visit reaches an ICT service. To do so, we consider data centres for what they are: *throughput* computing systems offering *ICT services* that can be *visited*. The notion of *throughput* is key because machines in data centres are not expected to perform *one specific action*, but to process *streams of actions*. Moreover, the notion of *service* is key, because simply *offering* the service, without any *visit*, is already power consuming. Moreover, offering a service over some time is energy consuming. Thus, the most relevant metric to optimise for might not only be the *energy-per-visit*, but rather the *power/Quality-of-service* trade-off. In other words, how much power is needed to offer a service with a given quality at any point in time.

What we propose and showcase in this paper is a test methodology and underlying testbed architecture, presented in Section II, where the *cyberspace* that one uses to access and visit a service, is reduced to a simple data centre that is made, in fact, of a single machine. This machine is offering services, and is subject to a stream of visits arriving at a given rate. While processing this stream, the power consumption is measured, among other system metrics. We then apply this methodology to four different hardware types, and two different services. Acknowledging the massive adoption of AI in daily usages, but also its large environmental impact [24], one of the services is an AI inference. The implementation of our methodology is described in Section III.

Measurement results are presented in Section IV. They show, in particular, how the power consumption of the machines is affected by the rate at which services are visited. We compile our take-aways in Section V. The numerous future research directions enabled by our methodology are discussed in Section VI. Section VII concludes.

2 Methodology

The experimental testbed shown on Figure 1 is in the centre of our test methodology. Its key component is the *service-under-test (SUT)*, which will handle requests sent remotely. The *SUT* is offered by a machine that we designate as the *hardware-under-test (HUT)*. These two components form the *offer* part of our methodology. As for the *consumption* of the service, the remote requests are sent in a controlled way by a *Load Tester Device (LTD)*. While the HUT processes the request stream, we measure its power consumption through a *Power Distribution Unit (PDU)*.

2.1 Offering the service

The SUT allows an arbitrary action to be triggered by a remote visit on the HUT, which will cause the latter to consume power. In our methodology, each visit to the SUT is independent, and the SUT is capable to serve a stream of such independent visits. To be aligned with modern ways of offering *web services*, very much like website visits, we assume the SUT to offer an HTTP endpoint. Optionally, one or more HTTP parameters can influence the action behaviour. A visit to the service is thus composed of an HTTP request, followed

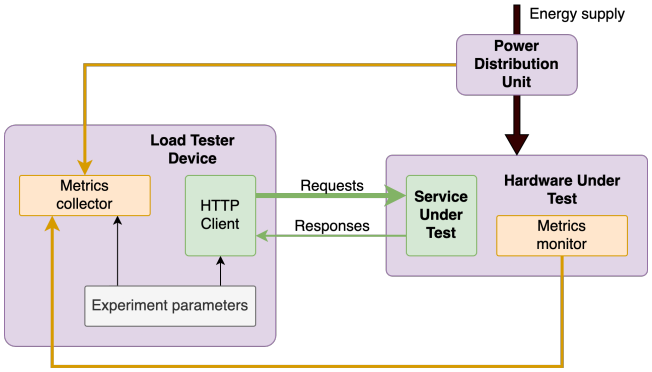


Figure 1: Experimental testbed setup

by an HTTP response. The visit is deemed completed when the response has successfully reached the request initiator, in our case the LTD, and when its content value has been validated to attest that the SUT has correctly processed the request. As stated above, the HUT aims to embody a machine located in a data centre, able to receive a stream of user requests.

The HUT must run two applications: (a) an implementation of one or several SUTs, and (b) a low-level hardware and OS metrics monitor, which records how the stream of requests affects the behaviour of the HUT while serving the requests.

As we are interested in effective energy consumption, we need the HUT power supply to be monitored. To obtain unchallengeable measurements, we assume that a calibrated *Power Distribution Unit (PDU)* is used. To ensure real-time and in-sync measurements, the PDU metrics must be accessible through an API, providing a fine-grained sampling rate. Alternative ways to measure power consumption include interrogating the *Baseboard Management Controllers (BMCs)*, typically using the IPMI protocol, or leveraging in-CPU mechanisms such as *RAPL* [29]. RAPL has been shown to yield accurate measurements [20], but is limited to a sub-domain of the system. By using a calibrated PDU, we avoid issues related to approximate *Baseboard Management Controller (BMC)* power measurements, and we capture the whole machine, as if deployed in the wild.

Note that, in our methodology, the SUT could fundamentally be offered by something else than a physical machine. It could be virtual machines (VMs), or containers. However, as we are focused on obtaining rigorous and effective power consumption numbers, and since power consumption of virtualised/containerised environments is not straightforward [6, 19, 27], we restrict ourselves to physical machines only.

Also note that we limit the measurements of the HUT to the machine itself only, thus not taking into account the power consumption of the network, nor the one of the infrastructure surrounding the machine (in particular, cooling). Such aspects will be considered in future work.

2.2 Consuming the service

The LTD is a computer used to conduct and lead the experiment. It has two roles: (a) to mock users that periodically visit the SUT,

and (b) to collect both low-level and power HUT metrics while the latter offers the SUT. Importantly, the LTD must be able to mock users concurrently. Moreover, the rate at which requests are emitted should be independent of the service latency, which is the time the service takes to send its response, applying an infinite population hypothesis.

3 Implementation of the methodology

3.1 *Hardware-under-test*

Table 1 lists the four machines that we used as HUTs. This selection aims to represent old and new machines alike, and also provides both ARM- and Intel-based architectures. As this paper focuses on the methodological approach rather than precise benchmarks, describing their complete hardware configurations is out of scope. Our approach also takes these machines *as is*, as if they would be rented as a *Metal-as-a-service*. However, we do note that the two Intel-based machines have the *Turbo Boost* and *Hyper-Threading* features turned on.

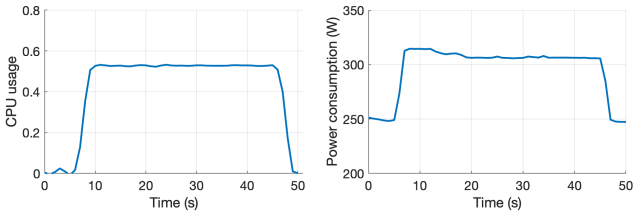
As for the low-level monitoring, we used the *Prometheus Node Exporter (PNE)* tool [28]. Once started, a PNE server exposes the requested real-time metrics on a given port through an API. To monitor the power consumption, we either used a *Riedo Network UPDU* [23] or an *Expert Power Control 1105* [16] metered PDU. Both system and power monitoring are sampled every second.

3.2 *Service-under-test*

We implemented two services. The first one is a synthetic benchmark consisting of a loop incrementing a counter until it reaches a limit provided as parameter. We call it the *Counting service* hereafter. The second one is a more practical one consisting of predicting the next token of a text provided as parameter using a pre-trained *BERT* model [7]. We designate it as the *Inference service*.

Both services are exposed by an *HTTP/1.1* web server using the *Java Spring Boot (JSB)* framework. JSB has been chosen due to its native support of multi-threads, a key factor to fulfil the concurrent users requirements.

We used a Python implementation of BERT published on *Hugging Face Hub* [5]. Pre-trained tokeniser and model are loaded on the `class` initialisation using the *Transformers* library [10], and are then used in a main `function` to encode the input as tokens, run the inference, and decode the predicted token. Note that we initially provided the *Inference service* using a Python application implemented with *FastAPI*, but Python's limitations in handling concurrency created a performance bottleneck. To circumvent this, our JSB implementation spawns a pool of Python processes. Upon arrival of an inference request, the JSB thread finds or waits for a free Python process, passes the text input through the *stdin* channel, and waits for the response to come back on the *stdout* channel. Our preliminary experiments have shown that using many parallel Python processes, each restricted to a single thread, yields better throughput (amount of requests per second) than using a single Python process with no threads restriction. Hence, in all our measurements, we start a number of Python processes equal to the cores (both physical and virtual) available on the HUT. This is not surprising, as sequential executions generally have a better yield than parallel ones. We stress that the details of this design are of minor importance, as we do not claim to faithfully represent a real-life service; our focus is on the benchmarking methodology.



(a) CPU usage **(b) Power consumption**

Figure 2: Sample of a experiment at 450 req/s on the QCT HUT using the *Counting service*

It is also worth noting that even though the QCT HUT has a strong GPU, we did not attempt to leverage this processing power to obtain comparable results throughout all the HUTs.

3.3 *Load Tester Device (LTD)*

We used the *k6* [21] load testing tool to generate the HTTP requests. K6 is designed to challenge web services by spawning so-called Virtual Users (VUs) accordingly, each of them sending requests to the service. VUs scheduling has been set to impose a *constant arrival rate* of requests during the whole test time, which is key to ensure that the effective load is unaffected by the service latency. If all the available VUs are busy waiting for a response, k6 creates more of them to comply with the requested load.

Nevertheless, reaching equilibrium, which is the adequate number of VUs to reach the requested load, takes a little time. This is why we configured k6 to monitor the last 20% of the test duration. We calculate the effective throughput by counting how many responses are received during the last fifth of the test time.

Within an experiment, the LTD proceeds as follows:

(1) Start recording the state of the HUT in a "cold" state for 10 seconds;
(2) Launch k6 to submit a configured load in *requests per second (req/s)* to a given service for 40 seconds;
(3) Wait for k6 to gracefully stop, waiting for responses of pending request to arrive;
(4) Wait for 5 more seconds to cool down the HUT.

During the whole experiment duration, the LTD periodically interrogates both the PNE and the PDU monitors and records all their relevant measurements.

Figures 2a and 2b illustrate a selection of measurements realised during a single experiment using the *Counting service*. In this particular case, 450 req/s have been sent to the QCT HUT. A clear correlation between the CPU usage, expressing which proportion of its total capacities is occupied, and the power consumption is shown, alongside with the presence of slopes going up and down accordingly to the load.

3.4 Remarks

It is worth noting that the behaviour of the HUT is influenced not only by the incoming requests sent to the SUT, but also by the PNE and the requests made to it. In other words, we disrupt the

Table 1: Characteristics of the machines used as HUTs

Machine	CPU	Frequency	Cores / Threads	Fabrication
QCT QuantaGrid S74G-2U	Nvidia Grace Hopper Superchip (ARM Neoverse V2)	3.3 GHz	72 / 72	2024
HPE ProLiant DL360 Gen9	2 x Intel(R) Xeon(R) E5-2640 v3	2.6 GHz	16 / 32	2015
Dell PowerEdge T430	2 x Intel(R) Xeon(R) E5-2620 v4	2.1 GHz	16 / 32	2015
Raspberry Pi 4 Model B	Broadcom BCM2711C0 (ARM Cortex-A72)	1.8 GHz	4 / 4	2021

measurements by measuring the system. However, this perturbation can be deemed negligible.

We also assume that the network connecting the HUT and the LTD has enough bandwidth to be considered "ideal" and thus does not act as a bottleneck of any kind.

4 Results

4.1 Performance, clock-speed and CPU usage

We first analyse at which maximum throughput each HUT can provide each SUT. Figure 3a shows that the throughput of the QCT HUT collapses above 800 req/s for both services. This is not a coincidence, as we calibrated the *Counting service* accordingly. Hence, we ask k6 to send $2.8e8$ as a parameter to the *Counting service* throughout this whole paper, noting that by doing so we achieve the same performance as a request on the *Inference service* on the QCT HUT. This HUT thus serves as the reference one. Note that we conducted our experiment on the *Inference service* using the *"hello"* parameter, which systematically gives *"again"* as a response. The parameter length did not significantly influence the *Inference service* performance, and such impact is out of this paper's scope. However, this subject could be addressed in future work.

By choosing this calibration, we are also acknowledging the fact that predicting one token with BERT is, given our implementation and our reference DUT, as complex as performing $2.8e8$ increments using a while condition. We can also make the following analysis: the QCT HUT features 72 cores clocked at ≈3.3 GHz, which thus collectively yield $72 \times 3.3e9 = 2.38e11\ cycles/s$. At critical load, this system sustains 800 $req/s \times 2.8e8\ loops/req = 2.24e11\ loops/s$, meaning that the machine reaches $2.38e11 \div 2.24e11 = 1.06\ cycle/loop$. Even though the while condition is reevaluated after each increment in our implementation, the compiler along with the CPU and its branch prediction are not far from the limit value of 1 cycle/count. Finally, we also conclude that one BERT inference uses around $2.8e8$ cycles given our implementation, which corresponds to approximately 300 million CPU cycles.

Regarding the HPE and Dell HUTs, they have distinct critical load limits for the two services with the *Counting service* collapsing first, as shown in Figures 3b and 3c. The inference certainly needs to wait on RAM reads, but these "lost" cycles can be compensated by Intel's Hyper-Threading feature and its virtual cores.

Finally, Figure 3d shows that the throughput of the *Inference service* on the Raspberry Pi collapses much earlier than the *Counting service*, probably due to its simplified architecture which can limit the amount of system inputs and outputs.

Note that in terms of cycles/loop, the two Intel-based HUTs need 1.14 cycle/loop, thus a little more than the Arm-based QCT HUT. In contrast, the Raspberry Pi requires 2.57 cycle/loop, denoting a much simpler CPU architecture.

Figure 4 shows that the request latencies are consistent with throughput measurements. The zero-load latency for the QCT HUT is just below 100 ms. We note that at 800 req/s, each core processes $11.\overline{1}$ req/s, which outputs one request every ≈90 ms. Regarding the *Inference service* on Intel HUTs, different regimes occur depending on whether virtual cores are used or not.

Figure 5 shows how the clock speed evolves. The individual dots represent individual samples, recorded while the HUT is loaded with requests. We clearly see the Dynamic Voltage and Frequency Scaling (DVFS) feature at play on Intel HUTs, and, to a lesser extent, on the Raspberry Pi. However, the clock speed of the QCT HUT remains stable.

Finally, Figure 6 shows how the CPU usage is affected by the load. The CPU usage increases almost linearly with the load for the *Counting service*, but such behaviour does not occur for the *Inference service*. Interestingly, Intel-based CPUs already reach their maximal capacity at 130 req/s, and remain at 100% until reaching their critical load. Again, this is likely due to the Hyper-Threading feature.

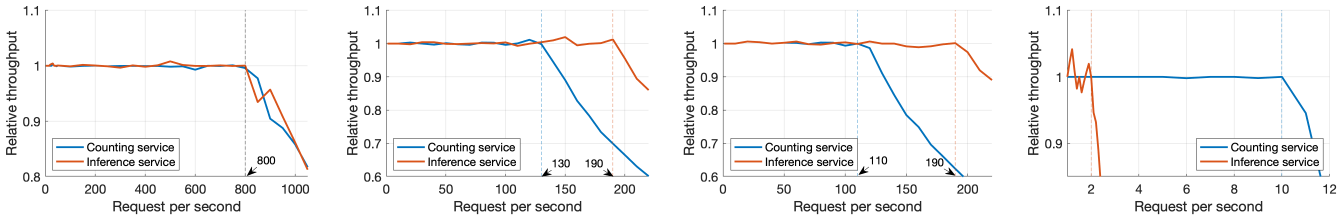
4.2 Power consumption

Figure 7 shows the power consumption. Consumption grows almost linearly with the load on all HUTs for the *Counting service*. However, the growth is clearly non-linear for the *Inference service*: except on the Raspberry Pi due to quick saturation, the power consumption plateaus to a maximum much before reaching the critical load.

Figure 8 relates the CPU usage to the power consumption, and clearly shows that the CPU usage is a *poor predictor* of the power consumption. This fact is particularly obvious on the QCT HUT: when the CPU indicates ≈60% usage, the power consumption reaches ≈320 W with the *Counting service*, but ≈470 W with the *Inference service*.

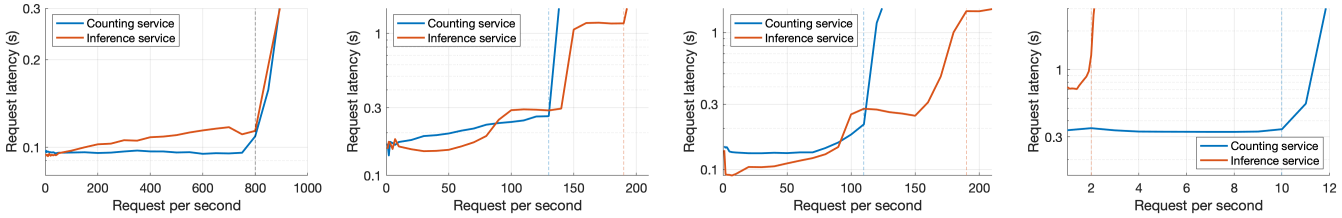
Other works have found that the power consumption scales almost linearly with the CPU usage on a data-streaming task [13]. Our experiment, in contrast, shows that the consumption of the whole machine varies depending on the nature of the load. This is the first important take-away of this paper: **the CPU usage is a poor indicator of the full machine power consumption**.

By dividing the power consumption, expressed in Watts thus Joules/s, by the number of requests per second, we obtain an energy consumption in Joules per request. Figure 9 shows that the energy cost of a BERT inference using our implementation can be as low as 0.64 J on the QCT HUT. Intel-based HUTs yield, in the best case, values between 1.13 and 1.67 J, and the Raspberry Pi drops up to 3.47 J.



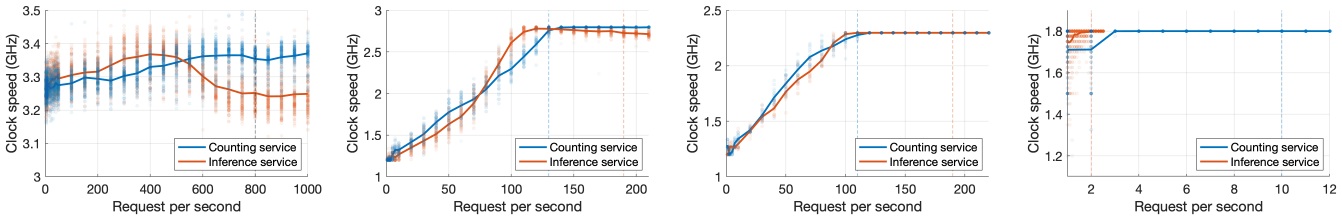
(a) QCT (b) HPE (c) Dell (d) Raspberry Pi

Figure 3: Proportion of successful requests (relative throughput)



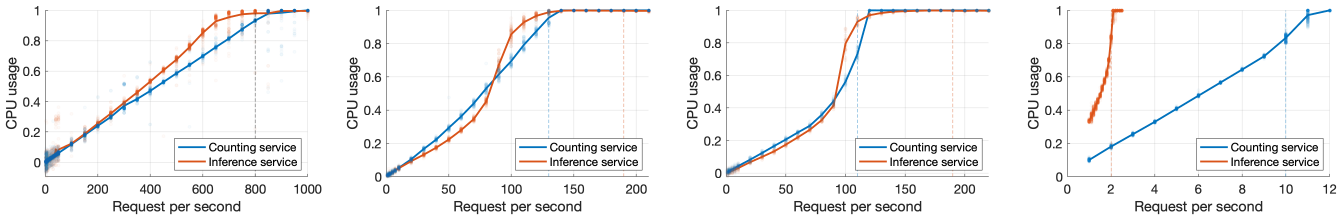
(a) QCT (b) HPE (c) Dell (d) Raspberry Pi

Figure 4: Measured average time taken to serve requests



(a) QCT (b) HPE (c) Dell (d) Raspberry Pi

Figure 5: Measured clock speed when serving requests



(a) QCT (b) HPE (c) Dell (d) Raspberry Pi

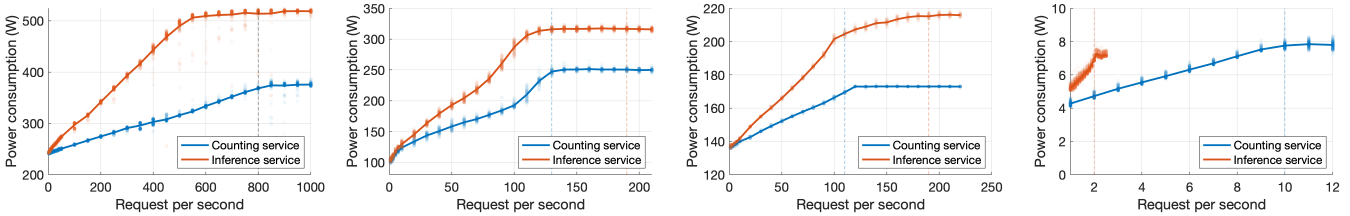
Figure 6: Measured relative CPU usage when serving requests

This represents, in our opinion, the second take-away from this paper: **an inference with BERT requires, in our setup, an operational energy in the Joules range**. This *bottom-up* measurement can be compared to *top-down* estimates suggesting that an OpenAI chatbot request consumes an energy within the Wh range [1], reaching 10 kJ. Additional work and analyses will be required to connect the two approaches. Notably, our BERT implementation is not optimised and does not use GPUs. Still, we found it interesting to be able to associate an energy quantity linked to an AI inference.

4.3 Impact of load on energy efficiency

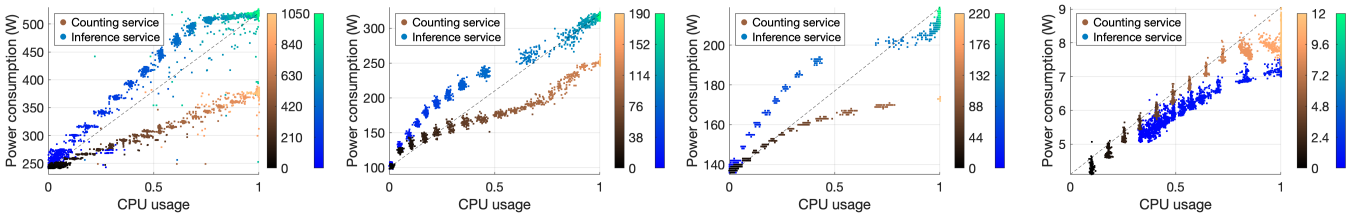
Figure 9 shows another interesting behaviour: the quantity of energy per request is heavily determined by the load sent to the HUTs.

Thus, the third take-away of this paper is that **associating an energy or a CO_2 quantity with a service visit implies that a strong hypothesis is made on the load the service is facing**. The load is actually more important than the hardware or software implementation of the service. The least energies-per-request,



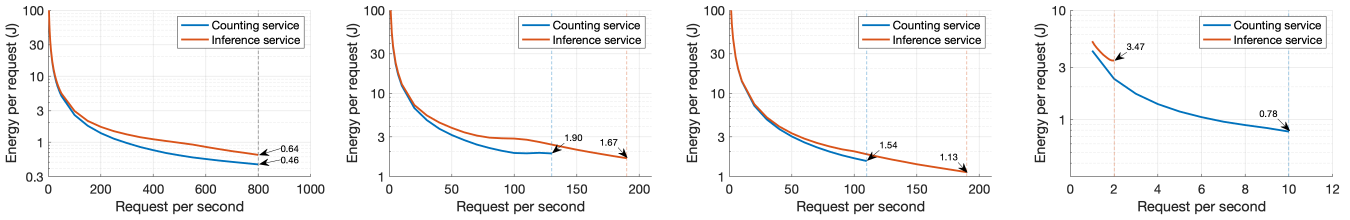
(a) QCT (b) HPE (c) Dell (d) Raspberry Pi

Figure 7: Measured power consumption



(a) QCT (b) HPE (c) Dell (d) Raspberry Pi

Figure 8: Measured power consumption with regards to CPU relative usage, the colour representing the load in req/s



(a) QCT (b) HPE (c) Dell (d) Raspberry Pi

Figure 9: Measured energy consumption per request

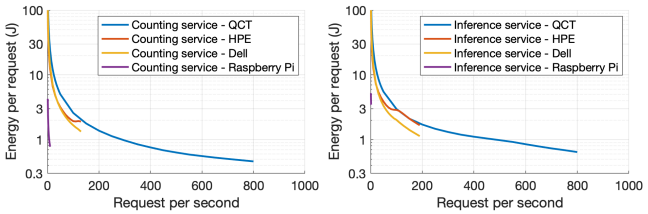
obtained at critical load, are shown on Figure 9. Notice that the 0.64 J/req delivered by the QCT HUT with the *Inference service* is less than one order of magnitude apart from the 3.47 J/req of the Raspberry Pi. In contrast, if one compares the range from 1 req/s to 800 req/s on the QCT HUT, the energy per request evolves from 0.64 J (at 800 req/s) to 100 J (at 1 req/s), representing more than 2 orders of magnitude.

In fact, it is key to realise that a service in real conditions should be optimised *for a given load*. Figure 10 shows that the QCT HUT, being the most modern one, outperforms the others in energy efficiency since it can reach energies per request as low as 0.64 J. Yet, this energy efficiency is available if and only if a sustained load of hundreds of requests per second is observed. What is striking in Figure 10 is that the QCT HUT is actually *dominated* by older HUTs for lower loads: the Raspberry Pi could be rated as inefficient with its 3 to 5 J per inference; but is actually the most efficient hardware if only 1 to 2 req/s are expected, that is, if compared in a fixed load context.

4.4 Power/QoS compromise

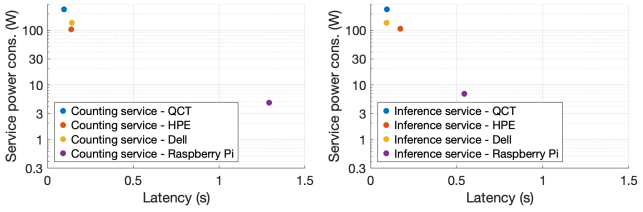
When considering a service offered on the web, the energy per request is clearly not the sole figure-of-merit to consider: a key aspect is also the *Quality-of-Service (QoS)*. If it is of uttermost importance to process requests in the shortest time possible, the service should be designed accordingly. Logically, we can expect speed to come at a power expense but, due to lower latency, not necessarily at an energy expense. Hence, when considering the offering of the service independently of the requests, we must speak in power terms. This *Power/QoS* trade-off is close to the *QoE/*CO_2 trade-off mentioned in Reference [17].

Figure 11 shows the different power/QoS compromises featured by the different HUTs with a load of 2 req/s. The QoS is measured in service response time, thus latency. We see that each HUT offers a different compromise. With the *Counting service*, the Dell HUT is dominated by the HPE HUT. With the *Inference service*, the QCT HUT is dominated by the Dell HUT. The final decision among non-dominated solutions on this *power/QoS trade-off* depends on the required *Service Level Agreement (SLA)* imposed to or by the service provider. If the service latency must be favoured (still assuming



(a) Counting service **(b) Inference service**

Figure 10: Services comparison on their energy usage



(a) Counting service **(b) Inference service**

Figure 11: Power/QoS compromises, assuming a load of 2 requests per second

a load of 2 req/s), the QCT (Counting service) or Dell (Inference service) HUTs are the obvious choices. Yet, if speed is not a concern but energy efficiency is, the Raspberry Pi offers the best compromise. This is our last take-away: **more modern machines do not yield better energy figures in all circumstances**, and can even be strictly Pareto-dominated in some contexts.

This way of representing the *power/QoS trade-off*, shown on Figure 11, permits to go beyond sole energy optimisations. Not only might one want to cut inefficiencies to be more energy conserving *at constant QoS*, but we might also consider *trading QoS against energy*.

5 Discussion

Let us summarise the above-mentioned take-aways:

(1) The CPU usage is a poor indicator of the full machine power consumption.
(2) An inference with BERT requires, in our setup, an operational energy in the Joules range.
(3) Associating an energy or a CO_2 quantity with a service visit implies that a strong hypothesis is made on the total load of the service.
(4) More modern machines do not yield better energy figures in all circumstances.

The first take-away brings an important conclusion: **CPU usage is a poor predictor of the entire machine consumption**. CPU usage is an easy metric to track, and it could be appealing, for instance, to measure the average usage of a system over some time to estimate its consumption. But in our case, such an estimation could be wrong by as much as 50%.

About the second take-away: we can affirm that when one pursues an optimisation goal, one must know the units and the order of magnitude of the figure-under-optimisation. While AI services flourish all around us, we found interesting to be able to quantise the burden of a widespread inference model. We argue that it would be beneficial for anyone developing an AI model to at least try to associate a figure of complexity and energy to each requested inference. In this paper, we measured complexity in a very simple way: we counted the number of loops needed for an inference (that is, 2.8*e*8), which, as shown above, is nearly equivalent to the number of CPU cycles, except for the Raspberry Pi. However, we plan to identify the FLOPS [9] or Instruction Per Second (IPS) figures in the future.

We cannot emphasise enough the third take-away, especially in an optimisation context. In our case, we could devote time and efforts to optimise our BERT implementation. Such an optimisation would be similar to compressing the media of a website to improve an arbitrary score like provided by *Website Carbon Calculator* [33]. But focusing on such tools to measure optimisation gains might divert us from much larger savings at the service provision level.

Finally, our forth take-away illustrates that modern machines seem to be optimised toward the south-east direction of the graphs on Figure 10, that is, to be more efficient at performing *more work*. But when *more efficient* is being multiplied by *more work*, the power consumption remains the same, yielding no sustainability gains. If one is only interested in optimising its services *at constant load* toward the south of the graphs exclusively, gains of modern architectures appear be more limited or even null, as our experiment has shown.

We see a relationship with the Jevons paradox, which predicts that the more energy-efficient something is made, the more it will be used, eliminating the absolute gain in energy efficiency. In our case, we show that more modern machines *must be* used more to become energy efficient, which is slightly different. But in both cases, there are no energy efficiency gains.

Another relationship can be made with the concept of strong- and weak-scaling in High Performance Computing [25]. Strong-scaling consists of realising the *same task* in *less time* with more parallelism. Such endeavour is difficult due to Amdahl's law, which is why one generally ends-up doing weak scaling, i.e. realising a *larger task* in the *same time* with more parallelism. Scaling down *energy* in absolute value might be as hard as scaling down *time*.

However, we cannot assume that loads on services will grow *forever*. We should start to look at ways to decrease power consumption *at constant load*, or even at decreasing loads. In other words, we need to tackle the *strong (down) scaling* of the power required by a service. For that matter, one has to acknowledge that the Raspberry Pi is a good example of sobriety, provided that one is willing to accept lower QoS.

The Raspberry Pi is, fortunately, not the only efficient way to offer a service for low loads: virtualisation and containerisation techniques are clearly others. These techniques allow large cost and energy reductions [18]. With regard to the angle of this paper, these techniques permit to distribute the burden of the *zero-load power consumption* over many VMs or containers, and thus to limit the J/req figure. However, one downside of VMs is that they cannot be as easily monitored as physical machines. Consequently, estimating the energy footprint of a service running on virtualised resources is challenging, especially without any information about

neighbouring VMs and/or underlying hypervisor and hardware. In Reference [18], authors measure how the power consumption of a service is affected by virtualisation. More precisely, they show that if a network intensive service is split across multiple identical VMs, consumption increases. But they do not tackle the estimation of the consumption imputable to *one specific VM* among multiple others executing *arbitrary* workloads.

6 Future Directions

The testbed we conducted only included a subset of all the variations that can be explored: we have identified several directions to explore that would be relevant for future work.

Consider VMs. As hinted above, many services undergoing low loads are nowadays provided using virtualised or containerised environments. We aim at reproducing our experiment using such *virtually hosted services*. One interesting aspect would be to identify the performance overhead inflicted by the hypervisor, brought by the virtualisation losses [3]. Network-related tasks can reach an overhead of up to 40% compared to bare-metal machines [30], but CPU intensive tasks does not introduce significant overhead [11]. But above all, confronting VMs to our methodology would hopefully allow us to better understand the energy consumption of VMs, and possibly to propose power allocation models.

Compare with RAPL or IPMI measurements. We plan to reproduce our experiment, but this time by collecting power measurements from the aforementioned alternative techniques: IPMI when available, and RAPL. We expect RAPL to be a much better predictor of the full machine power consumption than the CPU usage metric, but this assumption needs to be experimentally confirmed.

Include resource variations. Forcing hardware resource changes during an experiment, such as cutting access to one or multiple CPU cores or changing their frequencies, would provide interesting insights about the ability of machines to adapt to hardware failure or system overloads unrelated to the experiment. Thus, a more realistic quality of service offered in real-world setups could be measured.

The most interesting effect to measure and explore is the impact of the limitation of resources on the energy performance of the machines.

Impact of the software implementation on performance. Several software implementations, each providing the same API, could be compared to determine which technologies perform best in this specific setup. As for the *Counting service*, reaching less than 1 cycle per increment seems unlikely: we approximated that, except on the Raspberry PI, 1.06 to 1.14 cycles are needed to perform each increment, given our CPUs frequencies and a load of $2.8e8$ increments. If the service is challenged with smaller increments, the overhead associated with the API usage might grow. Implementation gains could be realised there.

In contrast, the *Inference service* seems to offer many more alleys for optimisation. We only performed basic optimisation when designing our implementations: there is clearly room for more.

Measure the network and data centre power consumption. Because such components are critical parts of ICT services, including them in our testbed would give more complete and interpretable insights on energy usage.

Serve more complex actions. We could introduce more complex services, such as tasks that challenge the RAM, reach to external resources using network connectivity, or serve web pages. This could lead to a reliable approximation of the Joules-per-visit figure for more common ICT services.

Adapt the testbed to other architectures. As the microservices architecture is nowadays largely adopted [8], the testbed could be adapted to fully integrate the characteristics related to its design specificities [12]. Moreover, an interesting addition would be to measure the energy overhead consumed by the multiple software instances running in such architectures.

Compare multiple architecture and sizes of AI models. AI models can have various characteristics that impact their energy consumption. This observation is particularly true for deep learning models, where their layered architecture adds up a large amount of operations, particularly in the field of generative AI. Our testbed could therefore include more models to be exposed as SUTs to determine which ones use the most energy to provide an inference. The amount of energy needed to collect and clean their dataset, alongside with their training phase, could also be explored.

Moreover, the complexity figure of such models should be measured in a more precise way, ideally in FLOPS or IPS, to successfully quantise their energy consumption.

Include GPUs. GPUs are most suited for parallel processing and large-scale computations, two important capacities that are very useful for, but not limited to, AI models. As machines now often include such hardware, such as the QCT one, it would be only rational to leverage their capabilities.

However, GPUs pose significant challenges. It is not straightforward to use them efficiently, and at full capacity [22]. Different methods of tasks scheduling [32] could be compared, as well as using both CPUs and GPU as a twin-system to serve AI inferences [34].

7 Conclusion

We proposed a test methodology and an associated testbed to measure the energy consumption of a computing system while offering and delivering a service. Our methodology allows to fine-tune the load imposed on the system by transposing the number of requests per second sent to different services-under-test. Our measurements, conducted on different hardware, allow us to unequivocally identify the energy associated with a visit.

We show that this latter figure is above all determined by the load imposed on the system. Without knowing this load, we posit that it is not possible to make a good estimation.

Our results also exhibit a weak-scaling effect on energy efficiency. Computers tend to become more energy efficient, but only when their loads are more demanding. We argue that focusing on energy efficiency to reach smaller consumptions at constant loads should be targeted in the future.

Acknowledgments

This research was partly supported by HES-SO through the project EIEIAE: *Évaluation des impacts environnementaux de l'intelligence artificielle en entreprise*.

References

[1] 2024. Powering Intelligence: Analyzing Artificial Intelligence and Data Center Energy Consumption. (2024). https://www.epri.com/research/products/000000003002028905

[2] International Energy Agency. 2024. Electricity 2024 – analysis - IEA. https://www.iea.org/reports/electricity-2024

[3] Jonatan Baumgartner, Christophe Lillo, and Sébastien Rumley. 2023. Performance losses with virtualization: Comparing bare metal to VMs and containers. In *International Conference on High Performance Computing*. Springer, 107–120.

[4] Andrew A Chien. 2023. GenAI: Giga ,TeraWatt-Hours,andGigaTonsofCO2. *Commun. ACM* 66, 8 (2023), 5–5.

[5] BERT community. 2024. google-bert/bert-base-uncased · Hugging Face. https://huggingface.co/google-bert/bert-base-uncased

[6] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. 2016. Data Center Energy Consumption Modeling: A Survey. *IEEE Communications Surveys & Tutorials* 18, 1 (2016), 732–794. https://doi.org/10.1109/COMST.2015.2481183

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] https://arxiv.org/abs/1810.04805

[8] Paolo Di Francesco, Patricia Lago, and Ivano Malavolta. 2019. Architecting with microservices: A systematic mapping study. *Journal of Systems and Software* 150 (April 2019), 77–97. https://doi.org/10.1016/j.jss.2019.01.001

[9] Romain Dolbeau. 2022. Correction to: Theoretical peak FLOPS per instruction set: a tutorial. *The Journal of Supercomputing* 78, 10 (July 2022), 12929–12929. https://doi.org/10.1007/s11227-022-04443-1

[10] Hugging Face. [n. d.]. *Transformers*. https://huggingface.co/docs/transformers/main/en/index Accessed: 2025-02-21.

[11] Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. 2014. An updated performance comparison of virtual machines and Linux containers. In *IBM research reports*. RC25482 (AUS1407–001). https://doi.org/10.1109/ISPASS.2015.7095802

[12] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. 2019. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, Providence RI USA, 3–18. https://doi.org/10.1145/3297858.3304013

[13] Joanna Georgiou, Moysis Symeonides, Michalis Kasioulis, Demetris Trihinas, George Pallis, and Marios D. Dikaiakos. 2022. BenchPilot: Repeatable & Reproducible Benchmarking for Edge Micro-DCs. In *2022 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, Rhodes, Greece, 1–6. https://doi.org/10.1109/ISCC55528.2022.9912882

[14] Michael Gillenwater. 2025. The differences between allocational and consequential greenhouse gas accounting —Summarized - GHG and Carbon Accounting, Auditing, Management & Training | Greenhouse Gas Management Institute. https://ghginstitute.org/2025/01/17/the-differences-between-allocational-and-consequential-greenhouse-gas-accounting-summarized/

[15] W3C Community Group. 2024. Home - Sustainable Web design. Retrieved Sept. 6, 2024 from https://sustainablewebdesign.org/

[16] GUDE Systems. [n. d.]. *Switched and metered LAN power socket: Expert Power Control 1105-1 - GUDE Systems*. https://gude-systems.com/en/products/expert-power-control-1105/

[17] Tobias Hoßfeld, Martín Varela, Lea Skorin-Kapov, and Poul E Heegaard. 2023. A Greener Experience: Trade-Offs between QoE and CO 2 Emissions in Today's and 6G Networks. *IEEE communications magazine* 61, 9 (2023), 178–184.

[18] Yichao Jin, Yonggang Wen, and Qinghua Chen. 2012. Energy efficiency and server virtualization in data centers: An empirical investigation. In *2012 Proceedings IEEE INFOCOM Workshops*. IEEE, 133–138.

[19] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A. Bhattacharya. 2010. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM Symposium on Cloud Computing* (Indianapolis, Indiana, USA) *(SoCC '10)*. Association for Computing Machinery, New York, NY, USA, 39–50. https://doi.org/10.1145/1807128.1807136

[20] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. 2018. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3, 2, Article 9 (March 2018), 26 pages. https://doi.org/10.1145/3177754

[21] Grafana Labs. [n. d.]. Grafana Labs k6 load tester. https://www.k6.io. Accessed: 2025-02-21.

[22] Jingyu Lee, Yunxin Liu, and Youngki Lee. 2021. ParallelFusion: Towards Maximum Utilization of Mobile GPU for DNN Inference. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning* (Virtual, WI, USA) *(EMDL'21)*. Association for Computing Machinery, New York, NY, USA, 25–30. https://doi.org/10.1145/3469116.3470014

[23] Riedo Networks Ltd. [n. d.]. Riedo networks ltd Universal power distribution units. https://www.rnx.ch/updu. Accessed: 2025-02-21.

[24] Sasha Luccioni, Yacine Jernite, and Emma Strubell. 2024. Power Hungry Processing: Watts Driving the Cost of AI Deployment?. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency* (Rio de Janeiro, Brazil) *(FAccT '24)*. Association for Computing Machinery, New York, NY, USA, 85–99. https://doi.org/10.1145/3630106.3658542

[25] Satoshi Matsuoka and Jens Domke. 2022. Life after Fugaku: What Have We Learned and How Do We Proceed as the End of Moore's Law Approaches?

[26] Mightybytes. [n. d.]. ecograder, website rating system. Retrieved Sept. 23, 2024 from https://ecograder.com

[27] Christoph Möbius, Waltenegus Dargie, and Alexander Schill. 2014. Power Consumption Estimation Models for Processors, Virtual Machines, and Servers. *IEEE Transactions on Parallel and Distributed Systems* 25, 6 (2014), 1600–1614. https://doi.org/10.1109/TPDS.2013.183

[28] Prometheus. [n. d.]. *GitHub - prometheus/node_exporter: Exporter for machine metrics*. https://github.com/prometheus/node_exporter Accessed: 2025-02-21.

[29] Guillaume Raffin and Denis Trystram. 2025. Dissecting the Software-Based Measurement of CPU Energy Consumption: A Comparative Analysis. *IEEE Trans. Parallel Distrib. Syst.* 36, 1 (Jan. 2025), 96–107. https://doi.org/10.1109/TPDS.2024.3492336

[30] Ryan Shea, Haiyang Wang, and Jiangchuan Liu. 2014. Power consumption of virtual machines with network transactions: Measurement and improvements. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 1051–1059. https://doi.org/10.1109/INFOCOM.2014.6848035

[31] Arman Shehabi, Sarah Josephine Smith, Alex Hubbard, Alexander Newkirk, Nuoa Lei, Md AbuBakar Siddik, Billie Holecek, Jonathan G Koomey, Eric R Masanet, and Dale A Sartor. 2024. *2024 United States Data Center Energy Usage Report*. Technical Report. Lawrence Berkeley National Laboratory. https://doi.org/10.71468/P1WC7Q

[32] Yuan Wen and Michael F.P. O'Boyle. 2017. Merge or Separate? Multi-job Scheduling for OpenCL Kernels on CPU/GPU Platforms. In *Proceedings of the General Purpose GPUs* (Austin, TX, USA) *(GPGPU-10)*. Association for Computing Machinery, New York, NY, USA, 22–31. https://doi.org/10.1145/3038228.3038235

[33] Scamper Ltd Wholegrain Digital. [n. d.]. Website Carbon Calculator. https://www.websitecarbon.com/. Accessed: 2025-02-21.

[34] Chengye Yu, Tianyu Wang, Zili Shao, Linjie Zhu, Xu Zhou, and Song Jiang. 2024. TwinPilots: A New Computing Paradigm for GPU-CPU Parallel LLM Inference. In *Proceedings of the 17th ACM International Systems and Storage Conference* (Virtual, Israel) *(SYSTOR '24)*. Association for Computing Machinery, New York, NY, USA, 91–103. https://doi.org/10.1145/3688351.3689164