*applied sciences*

MDPI

*Article*

# Energy Behavior of AI Workloads Under Resource Partitioning in Multi-Tenant Systems

**Jiyoon Kim [1], Siyeon Kang [1], Woorim Shin [1], Kyungwoon Cho [2] and Hyokyung Bahn [1,*]**

[1] Department of Computer Engineering, Ewha University, Seoul 03760, Republic of Korea; 2176101@ewhain.net (J.K.); kstar1029@ewhain.net (S.K.); wrim0923@ewhain.net (W.S.)
[2] Embedded Software Research Center, Ewha University, Seoul 03760, Republic of Korea; cezanne@ewha.ac.kr
* Correspondence: bahn@ewha.ac.kr

### Featured Application

This study provides empirical insights into AI workload energy behavior, supporting a paradigm shift in cloud pricing from resource allocation models to workload energy profiles.

### Abstract

Traditional cloud pricing models are allocation-centric, where users are charged based on reserved resources rather than workload energy consumption. However, modern AI workloads exhibit substantial and heterogeneous power behavior, limiting the effectiveness of such allocation-centric pricing. This paper presents a comprehensive experimental study of nine widely used workloads across 50 controlled configurations, including standalone and concurrent executions under varying resource partitions. Our results show that total system power is largely unaffected by how resources are divided among co-located workloads, except in cases of explicit resource under-provisioning or severe resource contention. Across 45 workload–core groups, 41 exhibit a coefficient of variation below 3% across different co-located workloads, demonstrating structural stability of workload-level power profiles under heterogeneous execution environments. In contrast, deployment choice (e.g., CPU versus GPU execution) can shift the same model into distinct power regimes. Based on measured power decomposition and scaling behavior, we derive an empirical categorization framework distinguishing GPU-dominant and CPU-dominant workloads, further characterized by utilization and memory dimensions. From an energy perspective, CPU utilization (for CPU-dominant workloads) and SM utilization (for GPU-dominant workloads) emerge as the primary determinants of power magnitude, while memory-related parameters contribute marginally to overall power. These findings provide empirical evidence that allocation-based pricing is a weak proxy for actual energy cost and motivate energy-aligned cloud management strategies grounded in workload power profiles. As our findings are derived from a controlled single-node experiment, evaluations under more realistic data center environments will be required for further generalization.

**Keywords:** cloud energy consumption; workload characterization; resource partitioning; AI workloads; energy-aware cloud billing

Check for updates

Academic Editor: Christos Bouras

Received: 3 March 2026
Revised: 21 March 2026
Accepted: 22 March 2026
Published: 24 March 2026

**Copyright:** © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

## 1. Introduction

Cloud infrastructures are traditionally billed based on allocated resources, such as the number of virtual CPUs, memory capacity, or attached GPUs [1]. Such allocation-

centric pricing models were originally designed to reflect infrastructure provisioning and capital expenditure (CAPEX) structures, where allocated resource units serve as a practical abstraction for hardware cost [2]. However, operational expenditure (OPEX), particularly energy and cooling costs, now constitutes a substantial and growing portion of data center expenses, especially in AI-intensive computing environments [3–5]. Accordingly, there is increasing interest in charging models that more accurately reflect energy usage rather than merely provisioned resource capacity [6,7].

However, total energy consumption depends not only on the amount of allocated resources but also on workload characteristics, hardware heterogeneity, and cross-resource interactions, making the construction of an accurate and fine-grained energy model challenging. AI workloads in particular exhibit distinct data-access patterns and computational intensity compared to traditional workloads, further complicating energy modeling [8,9]. Furthermore, in multi-tenant environments, the combined energy footprint of co-located workloads cannot be trivially inferred from resource shares alone [10,11]. Prior work on multi-tenant workload consolidation has primarily examined performance interference and resource contention, but its implications for system-level power behavior remain less explored [12,13].

Empirical observations reveal that workloads with the same nominal resources can exhibit order-of-magnitude differences in active power consumption [6]. A lightweight server process and a GPU-intensive matrix multiplication may occupy similar logical resources, yet impose radically different energy footprints. If energy consumption does not scale proportionally with allocated resource shares, then allocation-based charging may fail to reflect actual operational cost drivers in energy-conscious data center environments.

Existing research on energy-aware computing has largely focused on improving hardware efficiency, dynamic voltage and frequency scaling (DVFS), and energy-aware scheduling [14,15]. While these approaches address how to reduce energy consumption, they do not directly examine whether resource partitioning itself governs total system power in multi-tenant environments. In particular, the fundamental assumption that reallocating CPU or memory shares meaningfully changes overall power draw remains insufficiently tested.

This study provides an empirical examination of that assumption. We evaluate nine AI and non-AI workloads across 50 experimental cases, including standalone executions and 10 concurrent workload combinations under multiple CPU and memory allocation ratios. By measuring CPU, DRAM, GPU, and wall-level power simultaneously, we isolate how total system power responds to resource redistribution versus workload composition.

Our findings can be summarized in three central observations.

First, total system power exhibits allocation insensitivity. Across most workload combinations, redistributing CPU cores and memory capacity between co-located workloads produces negligible change in wall-level power. Once workload composition is fixed, overall power consumption remains effectively constant despite substantial shifts in resource allocation ratios, except in cases of explicit resource under-provisioning or severe resource contention. Resource allocation ratios therefore function as weak predictors of total system power under normal operating conditions.

Second, workloads preserve intrinsic power signatures under concurrency. CPU utilization patterns, GPU power levels, and scaling characteristics remain close to their standalone behavior even when executed alongside heterogeneous partners. This stability indicates that power behavior is largely workload-driven rather than interaction-driven, except in clearly identifiable resource-starvation scenarios.

Third, deployment mode fundamentally determines energy categorization. The same model executed on CPU versus GPU hardware may transition between entirely different

power regimes. Execution placement, rather than application label alone, governs whether a workload behaves as CPU-dominant or GPU-dominant in power terms.

Based on measured power decomposition and scaling characteristics, the evaluated workloads first separate into two primary energy-oriented groups: GPU-dominant and CPU-dominant. Within each group, further distinctions emerge along utilization and memory dimensions. CPU-dominant workloads can be differentiated by CPU utilization levels and memory footprint, while GPU-dominant workloads vary in streaming multiprocessor (SM) utilization and GPU framebuffer usage.

However, from an energy magnitude perspective, utilization consistently dominates over memory-related parameters. While memory footprint and framebuffer usage contribute to performance characteristics and may influence bottleneck behavior, their impact on overall power magnitude is comparatively limited. Instead, CPU utilization primarily determines power behavior within CPU-dominant workloads, and SM utilization plays the equivalent role within GPU-dominant workloads.

These findings indicate that workload power behavior is governed more by execution intensity than resource allocation ratios or memory footprint. If total system power is largely insensitive to resource partitioning—except under explicit under-provisioning or severe contention—yet strongly shaped by workload-driven utilization characteristics, then allocation-based billing cannot reliably reflect actual energy cost drivers. A workload-aware perspective is therefore necessary for designing energy-aligned charging and resource management strategies in AI-intensive cloud environments.

The contributions of this paper are summarized as follows.

- We analyze the power behavior of AI workloads under standalone and concurrent execution scenarios through a comprehensive experimental study.
- We identify that each workload exhibits a stable power profile that remains largely insensitive to co-located workloads and resource partitioning, except under explicit resource under-provisioning.
- We derive an energy-oriented categorization of workloads, providing a practical perspective for understanding energy behavior in multi-tenant execution environments.

The remainder of this paper is organized as follows. Section 2 describes the experimental setup, including system configuration, workload selection, and resource partitioning design. Section 3 presents empirical observations on power behavior under standalone and concurrent execution. Section 4 derives an energy-oriented workload categorization and discusses its implications for allocation-based cloud billing. Section 5 briefly discusses works related to this study. Finally, Section 6 concludes the paper.

## 2. Experimental Setup

### *2.1. System Configuration and Measurement Infrastructure*

All experiments were conducted on a single-socket server equipped with an Intel Xeon E5-2630 v4 processor (Intel Corp., Santa Clara, CA, USA) (10 cores with Hyper-Threading enabled), 32 GiB DDR4 2133 MHz memory (single DIMM configuration), and two NVIDIA Titan V (GV100) GPUs (NVIDIA Corp., Santa Clara, CA, USA). The operating system was installed on a Samsung 860 1 TB SSD (Samsung Electronics, Suwon, Republic of Korea, while workload data were stored on an HGST 4 TB HDD (HGST, Inc., San Jose, CA, USA). CPU frequency scaling was set to the ondemand governor with Turbo Boost disabled to ensure stable and reproducible power measurements.

Power consumption was measured at multiple levels to capture both component-level and system-level behavior. CPU and DRAM power were obtained via Intel RAPL interfaces [16], while GPU power and utilization metrics were collected using nvidia-

smi [17]. In addition, a wall-level external power meter was used to measure total system power consumption. Sampling was performed at 1 s intervals, which is sufficient to capture stable power characteristics of the workloads considered in this study. This granularity is also practical in cloud environments, as it can be obtained from standard system monitoring interfaces without additional overhead. Finer-grained sampling may capture more transient power fluctuations in highly bursty inference scenarios. However, such granularity is not necessary for the steady-state energy analysis targeted in this study.

To emulate multi-tenant cloud environments, virtual machine-like resource partitioning was implemented using Linux cgroup v2 (cpuset and memory.max) [18]. This approach allowed precise control over CPU core allocation and memory limits for concurrent workloads without introducing virtualization overhead. Each experiment followed a fixed protocol consisting of 30 s idle stabilization, 120 s workload execution, and 30 s cooldown, ensuring consistent measurement windows across all cases. In total, 50 experimental cases were evaluated, including 10 standalone executions and 10 concurrent workload combinations under four different CPU/memory allocation ratios.

To extract steady-state measurements, only the middle 120 s window (from 30 s to 150 s of each 180 s session) was used for analysis, excluding the initial idle stabilization and final cooldown phases. Prior to each experiment, the Linux page cache was flushed (drop_caches = 3) and the system was verified to have reached idle state, defined as both CPU and GPU utilization remaining below 5% for five consecutive seconds.

For data quality, sampling points with physically implausible values—such as negative RAPL readings or power values exceeding the system's rated capacity—were discarded. No additional statistical outlier filtering was applied; the arithmetic mean over the 100 s window naturally smooths transient fluctuations. The coefficient of variation (CoV), defined as $\sigma/\mu \times 100\%$, was computed across different resource allocation scenarios to quantify the sensitivity of wall power to resource partitioning.

### 2.2. Workloads

Nine workloads were selected to cover a broad spectrum of AI and non-AI computational behaviors, spanning GPU-intensive training and inference, CPU-bound computation, memory-sensitive language models, and lightweight server applications.

GPU-oriented workloads include GEMM, Training, ResNet-GPU, YOLO, and GPU-LLM. GEMM [19] performs continuous $4096 \times 4096$ single-precision matrix multiplications using PyTorch 2.5.1, representing a pure compute-bound GPU workload with near-maximum SM utilization. Training fine-tunes a pretrained ResNet-18 [20] on CIFAR-10 [21] with a batch size of 64, using SGD with a learning rate of 0.01 and momentum of 0.9, with two data-loader workers for preprocessing. ResNet-GPU performs inference on a pretrained ResNet-18 using batches of 32 randomly generated $3 \times 224 \times 224$ input tensors. YOLO runs YOLOv8-nano [22] in continuous video stream inference mode. GPU-LLM runs OPT-1.3B [23] in half-precision (fp16) inference, generating 128 tokens per prompt from a 512-token input context.

CPU-bound workloads include ResNet-CPU, LLM, ffmpeg, and NodeJS. ResNet-CPU performs the same ResNet-18 inference as ResNet-GPU but is restricted to CPU execution with a batch size of 8, isolating CPU-only energy behavior for a direct deployment comparison. LLM runs GPT-2 (124 M parameters) [24] on CPU, generating 50 tokens per prompt in an autoregressive loop. ffmpeg encodes synthetic 1080p video at 30 fps using the $\times$264 codec [25] with the medium preset, representing a realistic CPU-intensive media processing workload. NodeJS hosts a lightweight Express.js HTTP server (OpenJS Foundation, San Francisco, CA, USA) [26] that computes Fibonacci(35) and sorts a 10,000-element array per request, driven by 100 concurrent connections using autocannon.

Among the nine workloads, three serve as a deliberate reference anchors designed to establish clear energy boundaries: GEMM provides an upper bound for GPU power consumption through pure matrix multiplication, ResNet-CPU isolates CPU-only execution by intentionally bypassing GPU acceleration, and NodeJS represents a lightweight lower bound. The remaining six workloads including ffmpeg, Training, and LLM-GPU represent realistic production scenarios that validate whether the energy patterns observed at the anchor extend to practical applications.

For each workload, CPU utilization, GPU power, streaming multiprocessor (SM) activity, and memory usage were recorded, enabling detailed characterization of computational and power profiles.

### 2.3. Resource Partitioning Design

To analyze the effect of resource allocation ratios under concurrent execution, workload pairs were executed simultaneously under controlled CPU and memory partitions. Four allocation ratios were tested:

- 1:1 (10 cores/10 cores, 14 GB/14 GB)
- 2:1 (14 cores/6 cores, 20 GB/8 GB)
- 3:1 (15 cores/5 cores, 24 GB/4 GB)
- 5:1 (17 cores/3 cores, 24 GB/4 GB)

These ratios were selected to simulate practical cloud resource allocation asymmetries while maintaining total system capacity constant. Ten workload combinations were evaluated under all four ratios, producing 40 concurrent experiments. Together with the 10 standalone executions, this resulted in 50 total experimental cases. This design enables direct comparison between standalone and concurrent execution, as well as isolation of the impact of allocation ratios on system-level power behavior.

## 3. Power Behavior Under Resource Partitioning

This section presents the empirical power behavior observed under standalone and concurrent executions. We first examine the impact of CPU/memory allocation ratios on total system power, then analyze workload-level power profile stability under co-location, and finally investigate deployment-dependent energy shifts.

### 3.1. Allocation Insensitivity in Concurrent Execution

Before analyzing concurrent execution, we first examine the power profiles of each workload in isolation. Figure 1 presents the component-wise power decomposition for all standalone workloads, including the idle state. The idle power of each component is treated as a baseline, and the additional power consumed during workload execution is separately plotted.

As shown in Figure 1, the workloads can be broadly divided into two primary categories: GPU-dominant workloads (GEMM, LLM-GPU, Training, ResNet-GPU, and YOLO) and CPU-dominant workloads (ResNet-CPU, ffmpeg, LLM, and NodeJS). For GPU-dominant workloads, GPU power varies substantially across applications, ranging from approximately 33 W to 199 W, while CPU power remains relatively stable at around 12–14 W. In contrast, CPU-dominant workloads exhibit negligible GPU power but show considerable variation in CPU power, ranging from approximately 10 W to 33 W.

For both workload types, additional DRAM power remains modest even for workloads with relatively higher memory activity such as ffmpeg. Since a significant portion of DRAM power persists during idle states, the additional DRAM power reported here reflects only the dynamic increment above the baseline rather than the total memory power consumption. This indicates that memory power is largely static, persisting even without

intensive read/write activity [27]. These standalone results establish the intrinsic power structure of each workload, which serves as a reference for subsequent analysis under concurrent execution.

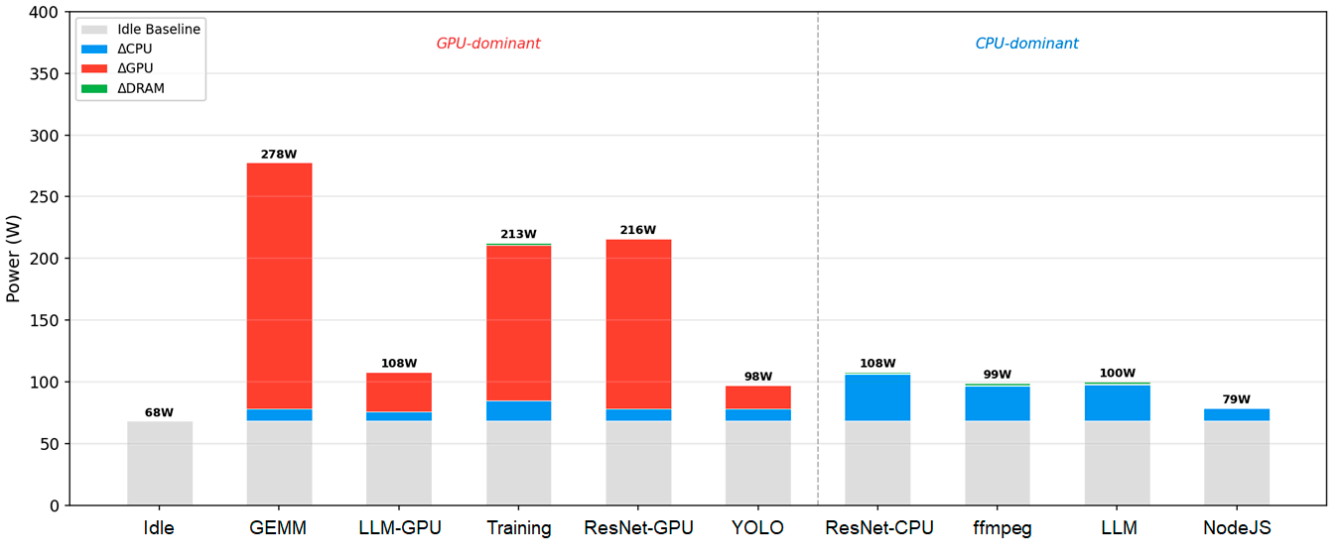



**Figure 1.** Component-wise power decomposition of standalone workload executions.

Having established the intrinsic power structure of standalone workloads, we now examine whether changing CPU and memory allocation ratios meaningfully affects total system power under concurrent execution. Table 1 summarizes the wall-level power measurements across 10 workload combinations. The selected combinations are not exhaustive but are chosen to represent typical interaction patterns across GPU-dominant, CPU-dominant, and lightweight workloads, enabling the observation of key resource contention behaviors while keeping the experimental study tractable. Experiments were conducted under four allocation ratios (1:1, 2:1, 3:1, and 5:1), and the table reports their maximum difference and variation. Throughout all experiments, thermally stable conditions were maintained to avoid thermal throttling and minimize cooling-related variations. For clarity, concurrent workload pairs are denoted by alphabetic labels (A–J), each corresponding to a fixed pair of workloads evaluated under varying allocation ratios. Specifically, A represents ResNet-CPU and GEMM, B represents ResNet-CPU and LLM, C represents GEMM and YOLO, D represents LLM and NodeJS, E represents ResNet-GPU and ResNet-CPU, F represents ResNet-GPU and LLM, G represents ffmpeg and LLM, H represents Training and ffmpeg, I represents LLM-GPU and GEMM, and J represents Training and NodeJS. These labels remain consistent throughout the analysis.

For nine out of the ten combinations, total wall power varied within ±1.4% despite substantial shifts in resource distribution. For example, in GPU + GPU combinations (e.g., combination I), wall power changed from 385.9 W (1:1) to 379.1 W (3:1), corresponding to only ±0.9% variation. Similarly, CPU + GPU combinations (e.g., combination A) exhibited changes from 372.4 W to 361.8 W, remaining within ±1.4%. The observed variations remain consistently small across allocation ratios, as indicated by low coefficients of variation and limited maximum differences. Although inferential statistical testing was not performed, these results provide evidence of allocation insensitivity from a practical perspective.

**Table 1.** Wall-level power under concurrent execution across resource allocation ratios.

| | Workloads | Type | 1:1 (W) | 2:1 (W) | 3:1 (W) | 5:1 (W) | Maximum Differences (W) | Maximum Variations |
|---|---|---|---|---|---|---|---|---|
| A | ResNet-CPU + GEMM | CPU + GPU | 372.4 | 367.0 | 365.1 | 361.8 | 10.6 | ±1.4% |
| B | ResNet-CPU + LLM | CPU + CPU | 175.2 | 173.1 | 172.0 | 171.1 | 4.1 | ±1.2% |
| C | GEMM + YOLO | GPU + GPU | 370.9 | 363.4 | 362.2 | 362.1 | 8.8 | ±1.2% |
| D | LLM + NodeJS | CPU + CPU | 166.6 | 165.4 | 164.7 | 164.9 | 1.9 | ±0.6% |
| E | ResNet-GPU + ResNet-CPU | GPU + CPU | 293.5 | 299.6 | 298.6 | 292.5 | 7.1 | ±1.2% |
| F | ResNet-GPU + LLM | GPU + CPU | 297.0 | 300.0 | 298.2 | 296.2 | 3.8 | ±0.6% |
| G | ffmpeg + LLM | CPU + CPU | 170.0 | 170.8 | 170.0 | 171.1 | 1.1 | ±0.3% |
| H | Training + ffmpeg | GPU + CPU | 240.8 | 293.7 | 295.3 | 297.7 | **56.9** | **±10.1%** |
| I | LLM-GPU + GEMM | GPU + GPU | 385.9 | 380.6 | 379.1 | 380.3 | 6.8 | ±0.9% |
| J | Training + NodeJS | GPU + CPU | 295.9 | 288.2 | 294.4 | 294.6 | 7.6 | ±1.3% |

These results indicate that once the workload pair is fixed, total system power is largely invariant to how CPU cores and memory capacity are partitioned between them. In other words, energy consumption is predominantly determined by workload composition rather than resource allocation ratio.

An exception was observed in one specific combination (combination H), where wall power varied significantly across allocation ratios. Further analysis in the following Section 3.2 indicated that this deviation resulted from a CPU resource bottleneck caused by insufficient allocation, which constrained the data-loading pipeline of a GPU-dominant training workload and reduced GPU utilization. This boundary case suggests that allocation insensitivity holds as long as supporting pipeline resources are provisioned sufficiently for normal execution.

Overall, the experimental evidence suggests that resource partitioning does not serve as a reliable proxy for system-level energy consumption under multi-tenant execution.

### *3.2. Preservation of Workload Power Profiles Under Concurrency*

We next examine whether individual workloads maintain their intrinsic power characteristics when executed concurrently. Tables 2 and 3 compare standalone and concurrent execution profiles for CPU utilization and GPU power, respectively. Across 20 workload instances within the 10 combinations, 19 maintained CPU utilization within ±3 percentage points of their standalone values when executed under a 1:1 resource allocation ratio. Similarly, GPU power levels remained close to standalone measurements. For example, GEMM exhibited 224.1 W under standalone execution, and 223.7 W (combination A), 229.2 W (combination C), and 227.8 W (combination I) under concurrency (within ±5.1 W of standalone).

The only substantial deviation was observed for the Training workload in combination H. Under standalone execution, Training consumed 151.2 W of GPU power, whereas under concurrency it dropped by 51.9 W to 99.3 W. This discrepancy was not due to a change in the intrinsic workload profile, but rather to a CPU bottleneck in that specific scenario. The underlying cause is further illustrated through the additional results presented in the following figures.

Figures 2 and 3 illustrate how CPU utilization and GPU SM utilization change when selected CPU-intensive and GPU-intensive workloads are executed either standalone or concurrently with different partner workloads. The *x*-axis represents the number of allocated CPU cores, which correspond to 1:1, 2:1, 3:1, and 5:1 resource allocation scenarios explained in Section 2.3.

**Table 2.** Profile preservation under concurrency: standalone vs. concurrent CPU utilization.

| Combination | Workload 1 | CPU Utilization | | | Workload 2 | CPU Utilization | | |
|---|---|---|---|---|---|---|---|---|
| | | Standalone | Concurrent | Δ | | Standalone | Concurrent | Δ |
| A | ResNet-CPU | 46.0% | 45.2% | −0.8 | GEMM | 5.0% | 5.0% | 0 |
| B | ResNet-CPU | 46.0% | 43.0% | −3.0 | LLM | 44.8% | 44.4% | −0.4 |
| C | GEMM | 5.0% | 5.0% | 0 | YOLO | 5.6% | 5.6% | 0 |
| D | LLM | 44.8% | 44.7% | −0.1 | NodeJS | 5.1% | 5.0% | −0.1 |
| E | ResNet-GPU | 5.0% | 4.8% | −0.2 | ResNet-CPU | 46.0% | 43.6% | −2.4 |
| F | ResNet-GPU | 5.0% | 5.0% | 0 | LLM | 44.8% | 45.2% | 0.4 |
| G | ffmpeg | 35.3% | 36.3% | 1 | LLM | 44.8% | 44.3% | −0.5 |
| H | Training | 6.5% | 3.4% | −3.1 | ffmpeg | 35.3% | 34.7% | −0.6 |
| I | LLM-GPU | 4.2% | 4.2% | 0 | GEMM | 5.0% | 5.0% | 0 |
| J | Training | 6.5% | 6.6% | 0.1 | NodeJS | 5.1% | 5.1% | 0 |

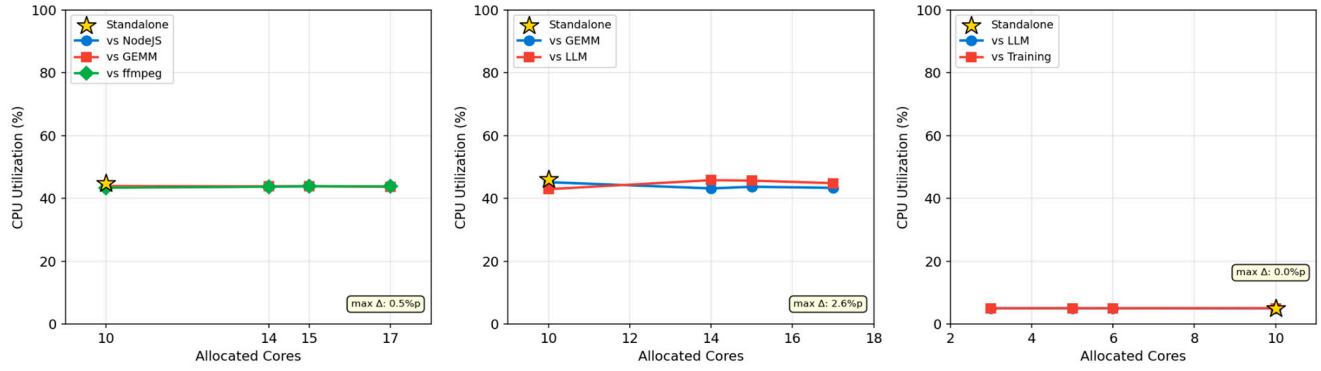
**Table 3.** Profile preservation under concurrency: standalone vs. concurrent GPU power.

| Combination | Workload 1 | GPU Power (W) | | | Workload 2 | GPU Power (W) | | |
|---|---|---|---|---|---|---|---|---|
| | | Standalone | Concurrent | Δ | | Standalone | Concurrent | Δ |
| A | ResNet-CPU | N/A | N/A | N/A | GEMM | 224.1 | 223.7 | −0.4 |
| B | ResNet-CPU | N/A | N/A | N/A | LLM | N/A | N/A | N/A |
| C | GEMM | 224.1 | 229.2 | 5.1 | YOLO | 43.9 | 42.2 | −1.7 |
| D | LLM | N/A | N/A | N/A | NodeJS | N/A | N/A | N/A |
| E | ResNet-GPU | 162.8 | 146.6 | −16.2 | ResNet-CPU | N/A | N/A | N/A |
| F | ResNet-GPU | 162.8 | 153.2 | −9.6 | LLM | N/A | N/A | N/A |
| G | ffmpeg | N/A | N/A | N/A | LLM | N/A | N/A | N/A |
| H | Training | **151.2** | **99.3** | **−51.9** | ffmpeg | N/A | N/A | N/A |
| I | LLM-GPU | 57.4 | 56.2 | −1.2 | GEMM | 224.1 | 227.8 | 3.7 |
| J | Training | 151.2 | 155.1 | 3.9 | NodeJS | N/A | N/A | N/A |

For example, Figure 2a shows the CPU utilization of the LLM workload under standalone execution and when co-located with NodeJS, GEMM, and ffmpeg. As shown in the figure, CPU utilization remains effectively unchanged regardless of the partner workload or standalone execution. Figure 2b,c shows similar results for other CPU-intensive workloads, including ResNet-CPU and NodeJS. In both cases, CPU utilization remains stable under standalone and concurrent execution.

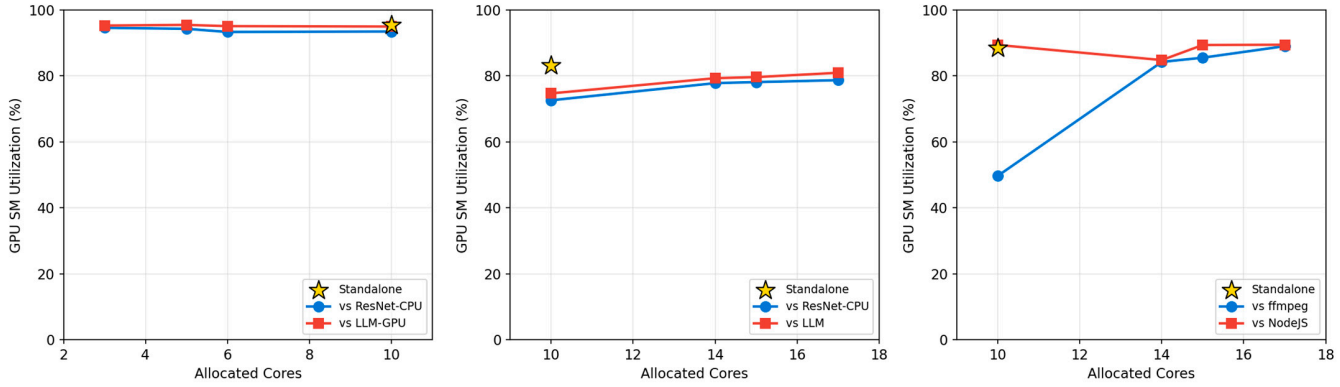
Figure 3 presents SM utilization patterns for GPU-intensive workloads under standalone execution and co-location with different partner workloads. For instance, Figure 3a shows GEMM executed alone and alongside ResNet-CPU and LLM-GPU. In all cases, GPU utilization remains nearly identical. Similar behavior is observed for ResNet-GPU in Figure 3b.

The only notable deviation occurred in Figure 3c, specifically when the Training workload was paired with ffmpeg. When 10 CPU cores were allocated to Training under this pairing, SM utilization dropped sharply from approximately 89% to around 50%, indicating that the CPU data-loading pipeline had become a bottleneck. This boundary case is consistent with the data stall phenomenon identified in prior work, where insufficient CPU resources for data preprocessing directly constrain GPU utilization during DNN training [28].




(**a**) LLM (**b**) ResNet-CPU (**c**) NodeJS

**Figure 2.** CPU utilization of CPU-intensive workloads under standalone and concurrent execution across varying CPU core allocations.




(**a**) GEMM (**b**) ResNet-GPU (**c**) Training

**Figure 3.** GPU SM utilization of GPU-intensive workloads under standalone and concurrent execution across varying CPU core allocations.

This behavior did not occur when Training was executed standalone or paired with a lightweight workload such as NodeJS. Under those conditions, even with 10 allocated cores, SM utilization remained stable at approximately 89%. This confirms that the observed degradation was not due to the absolute number of CPU cores, but rather to contention with a CPU-intensive co-located workload. This observation explains the behavior of combination H in Table 3. Once sufficient CPU resources were restored in the ffmpeg pairing ($\geq$14 cores), SM utilization returned to its expected range.

The transition to a resource-starvation regime is identified when the GPU workload can no longer sustain its standalone execution profile due to insufficient CPU pipeline capacity. In such cases, starvation manifests as a clear reduction in GPU SM utilization relative to the standalone baseline. This transition depends on workload characteristics and system configuration and cannot be represented by a universal numerical threshold. For example, the starvation case observed in combination H was accompanied by an SM utilization of roughly 50%, whereas YOLO and LLM-GPU exhibit stable behavior even when SM utilization is below 30% (see Section 4.1). Despite this, workload-level energy consumption can still be consistently determined from the observed resource utilization.

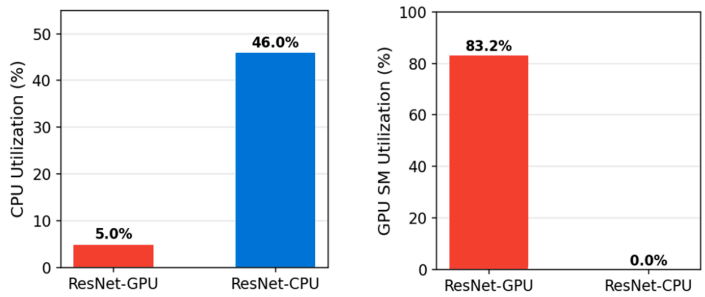
The deviation therefore reflects a system-level bottleneck under co-location with a CPU-intensive workload, likely related to shared pipeline or memory subsystems, rather than a change in the intrinsic power profile of the Training workload. It is associated with admission-level resource provisioning decisions, not with workload-driven power

characteristics. Taken together, these findings indicate that power behavior is primarily workload-specific and remains robust across realistic concurrent execution scenarios.

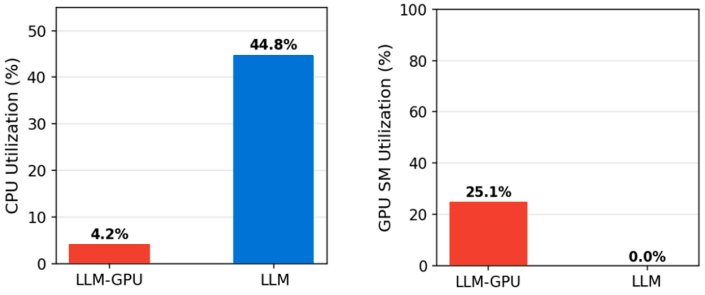
### *3.3. Deployment-Dependent Energy Shifts*

Beyond allocation ratios and concurrency effects, we observe that deployment choice (CPU versus GPU execution) fundamentally alters workload energy categorization. ResNet provides a representative example among our evaluated workloads. Although both ResNet-CPU and ResNet-GPU execute the same ResNet-18 model architecture, their power behavior differs substantially. As shown in Figure 1, CPU-based deployment consumed approximately 108 W, whereas GPU-based deployment consumed 216 W under identical system conditions.

To further quantify this difference, Figure 4a compares CPU utilization and GPU SM utilization of the ResNet workload when deployed in the two execution modes. As illustrated in the figure, CPU-based inference exhibited approximately 46% CPU utilization with no GPU activity, clearly placing it in the CPU-dominant regime. In contrast, GPU deployment resulted in approximately 83% GPU utilization and only about 5% CPU utilization, shifting it into a GPU-dominant class.




(**a**) ResNet-GPU vs. ResNet-CPU




(**b**) LLM-GPU vs. LLM

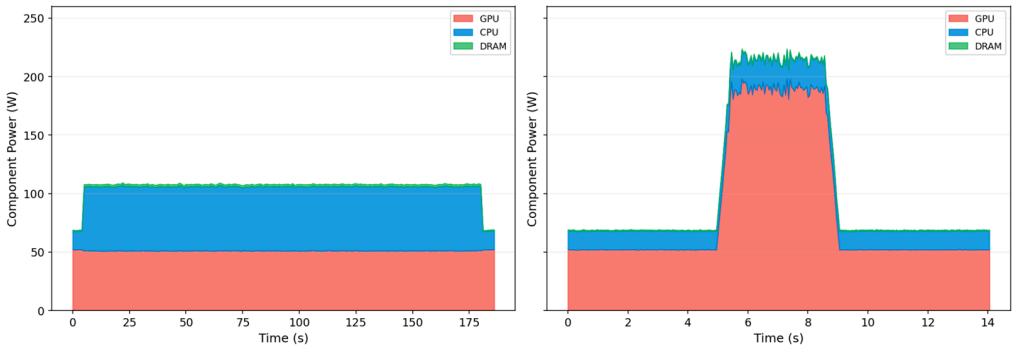
**Figure 4.** CPU utilization and GPU SM utilization of the ResNet and LLM workloads under CPU and GPU deployment modes.

This deployment-dependent divergence demonstrates that energy classification cannot rely solely on application identity. Execution mode must be explicitly considered, as the same algorithm may occupy entirely different positions in the system power spectrum depending on hardware mapping. Our results reveal an almost twofold difference in power consumption between CPU and GPU deployments, reinforcing that power impact is governed by execution characteristics rather than nominal resource assignment.

Additionally, a comparable pattern is observed across language model workloads, as shown in Figure 4b. The CPU-based language model (GPT-2) consumes 44.8% CPU utilization with no GPU involvement, whereas the GPU-deployed LLM utilizes 25.1% GPU SM with minimal CPU usage (4.2%). While these involve different model architectures and thus do not constitute a direct deployment comparison, they illustrate that language

model workloads can fall into fundamentally different energy categories depending on execution hardware.

Meanwhile, the impact of deployment on the power profile does not directly translate to energy consumption, as the two metrics are fundamentally different in nature. Figure 5 shows the measured power profiles over time while performing the same inference task on 10,000 input images using ResNet-18. As shown in the figure, ResNet-GPU exhibits an instantaneous power of approximately 216 W, which is about twice that of ResNet-CPU (108 W). However, the elapsed execution time differs substantially: ResNet-GPU completes the inference in 3.1 s, whereas ResNet-CPU requires 175.4 s. As a result, when energy consumption is considered, the relationship is reversed. Despite the higher instantaneous power, the significantly shorter execution time of ResNet-GPU leads to substantially lower energy consumption—approximately 96.5% less energy per inference than ResNet-CPU.




(**a**) ResNet-CPU (**b**) ResNet-GPU

**Figure 5.** Comparison of ResNet-CPU and ResNet-GPU in terms of power profile and energy consumption.

These results indicate that execution efficiency is fundamentally different from instantaneous power magnitude. While GPU execution exhibits higher power draw, it achieves significantly higher throughput, resulting in lower overall energy consumption.

## 4. Energy-Oriented Workload Categorization and Implications

The empirical patterns observed in Section 3 indicate that power behavior in multi-tenant environments is fundamentally workload-driven rather than allocation-driven. Building on these observations, we derive an energy-oriented workload categorization grounded in measured power structure, utilization characteristics, and scaling behavior.

### *4.1. Derivation of Energy Categories*

The most fundamental energy-based distinction among the nine evaluated workloads is whether system power is primarily driven by the GPU or the CPU. This binary separation establishes the first and most significant axis of energy-oriented categorization.

Figure 6 reveals a clear separation between GPU-dominant and CPU-dominant workloads by showing GPU SM utilization and CPU utilization. GPU-dominant workloads are characterized by high GPU power draw and substantial SM utilization, while CPU usage remains comparatively low and relatively uniform. In contrast, CPU-dominant workloads exhibit significant CPU utilization with negligible GPU contribution.

This primary distinction aligns with the dominant source of active power and therefore represents the most influential factor in determining overall system power behavior.

Within the GPU-dominant group, further structure emerges as shown in Figure 7. Arithmetic-intensive workloads such as GEMM, ResNet-GPU, and CNN Training exhibit high SM utilizations of 83–95%. These workloads form an SM-heavy subcategory and define the upper bound of the system energy spectrum.

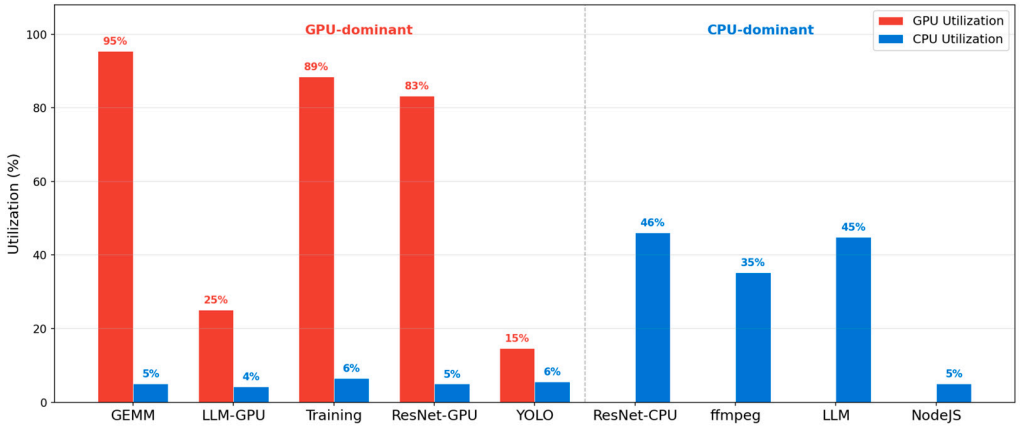



**Figure 6.** CPU utilization and streaming multiprocessor (SM) utilization across nine workloads.

By contrast, LLM-GPU and YOLO exhibit substantially lower SM activity (approximately 15–25%). A second orthogonal dimension further differentiates GPU workloads: framebuffer memory usage. From this perspective, LLM-GPU shows the largest memory usage, exceeding 2600 MiB, indicating a memory-intensive execution pattern. CNN Training also exhibits high framebuffer usage (approximately 2200 MiB), whereas ResNet-GPU, GEMM, and YOLO remain within a relatively modest range of 400–600 MiB.

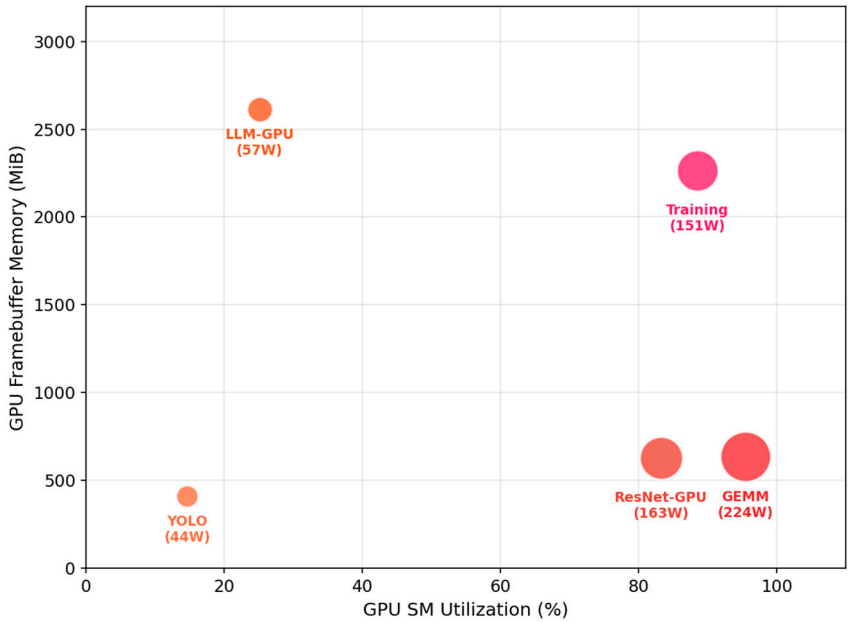



**Figure 7.** Internal spectrum of GPU-dominant workloads: SM-heavy vs. memory-heavy. Bubble size indicates GPU power consumption.

Considering these two dimensions jointly—SM utilization and framebuffer memory usage—distinct internal categories emerge. Training uniquely combines high SM utilization and high memory usage, reflecting its backward-pass computation and gradient storage requirements. YOLO occupies the lowest position in both dimensions, with approximately 15% SM utilization and 411 MiB of framebuffer usage, representing a comparatively lightweight GPU workload. ResNet-GPU and GEMM are clearly SM-heavy, whereas LLM-GPU is primarily memory-heavy.

This two-dimensional spectrum demonstrates that GPU-dominant workloads cannot be treated as a monolithic class; their internal energy structures vary substantially depending on the balance between computational intensity and memory footprint. From

the perspective of power profiling, however, SM utilization exerts a far stronger influence on overall GPU power draw than framebuffer memory usage. While memory-intensive workloads may increase memory occupancy and potentially introduce performance bottlenecks, their impact on total power magnitude is comparatively limited. In practical terms, GPU utilization—particularly sustained SM activity—serves as the principal determinant of the power profile within the GPU-dominant category.

CPU-dominant workloads also exhibit heterogeneous power behavior, as illustrated in Figure 8. The figure plots CPU utilization against RSS memory footprint, with bubble size representing CPU power consumption. ResNet-CPU, ffmpeg, and LLM demonstrate relatively high CPU utilization levels (approximately 35–46%), and correspondingly consume higher CPU power in the range of 45–54 W. In contrast, NodeJS exhibits minimal CPU utilization (approximately 5%) and correspondingly low CPU power (approximately 27 W). This pattern indicates a strong positive relationship between CPU utilization and CPU power magnitude within the CPU-dominant group.

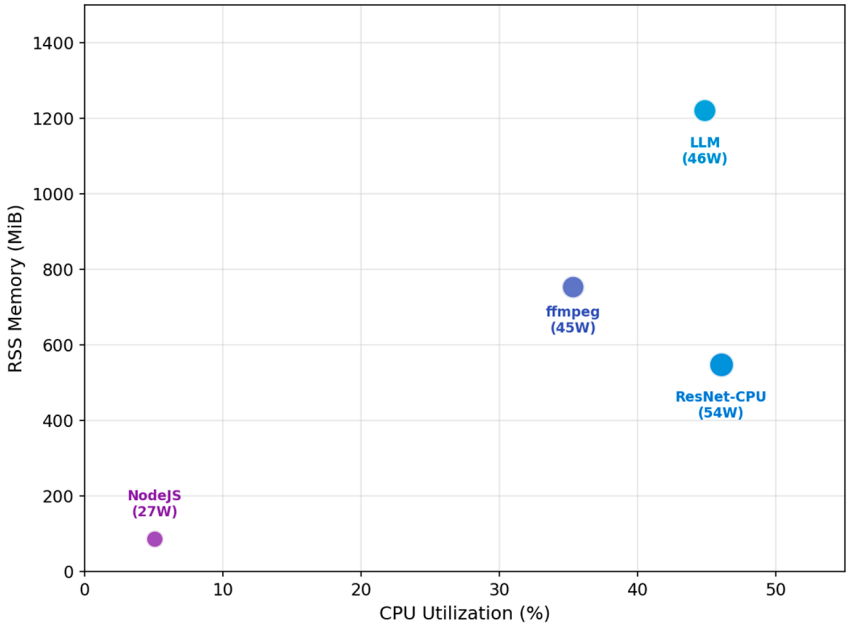



**Figure 8.** Internal spectrum of CPU-dominant workloads: compute-bound vs. memory bound. Bubble size indicates CPU power consumption.

From a memory perspective, LLM exhibits the largest RSS footprint (approximately 1220 MiB), reflecting its memory-intensive execution characteristics. However, despite substantial differences in memory footprint, CPU power remains comparable across LLM, ffmpeg, and ResNet-CPU, even though ffmpeg and ResNet-CPU operate with considerably smaller RSS usage (approximately 550–760 MiB). This observation suggests that RSS memory footprint has limited influence on overall CPU power magnitude compared to CPU utilization.

NodeJS exhibits both minimal CPU utilization and minimal RSS footprint (approximately 87 MiB), reinforcing its characterization as a lightweight workload with low energy demand across system components.

Taken together, the nine evaluated workloads empirically cluster into two primary energy-oriented categories—GPU-dominant and CPU-dominant—with lightweight workloads occupying the lower bound of the energy spectrum. Within each dominant category, subtypes emerge based on computational intensity and memory characteristics; however, from an energy magnitude perspective, utilization remains the principal determinant of

power behavior. These categories arise from measured power structure and microarchitectural characteristics rather than application-level labels.

From an operational perspective, these categories can be approximated using simple utilization-based criteria. Workloads with non-negligible SM utilization can be treated as GPU-dominant, whereas those with near-zero SM utilization can be regarded as CPU-dominant. From an energy attribution perspective, however, strict categorical separation is not essential. Energy contribution can be directly determined from the observed utilization of each resource. Even in cases of GPU-dominant workloads, both SM utilization and CPU utilization are observable and can be jointly used for energy attribution, without requiring strict categorical separation.

### *4.2. Implications for Energy-Aware Cloud Charging*

The findings of this study have direct implications for cloud resource management and pricing design. Traditional cloud billing models are allocation-centric, charging users based on reserved virtual CPUs, memory capacity, or attached GPUs. However, the experimental evidence presented here demonstrates that total system power under concurrent execution is largely insensitive to allocation ratios, except under explicit resource under-provisioning. Moreover, workload-level power signatures remain structurally stable across heterogeneous co-location scenarios. This indicates that power behavior is intrinsic to workload characteristics rather than determined by proportional resource shares.

Because category membership is preserved under realistic multi-tenant execution, energy-oriented workload classes are not artifacts of isolated benchmarking but reflect stable structural properties. Moderate resource asymmetries do not alter category boundaries, reinforcing the practical applicability of workload-based categorization in operational cloud environments.

An energy-aware charging framework may therefore shift emphasis from resource reservation metrics to workload-driven characteristics. At the highest level, distinguishing between GPU-dominant and CPU-dominant workloads captures the primary axis of energy magnitude. However, workload type alone is insufficient for precise energy attribution. Our results further indicate that, within each dominant category, utilization metrics serve as the principal determinants of power behavior. Specifically, CPU utilization governs energy magnitude in CPU-dominant workloads, while streaming multiprocessor (SM) utilization plays the equivalent role in GPU-dominant workloads. In contrast, memory-related parameters—such as RSS footprint or GPU framebuffer usage—exert comparatively limited influence on overall power magnitude.

In practical cloud environments, aggregate CPU and GPU power can be readily measured at the system level, yet disaggregating total energy consumption across co-located workloads remains challenging. The present findings suggest that once workload category is identified, workload-level utilization statistics—CPU utilization for CPU-dominant workloads and SM utilization for GPU-dominant workloads—provide a tractable basis for approximating per-workload energy contribution. These utilization metrics are already observable through standard system monitoring interfaces, making workload-aware energy accounting operationally feasible. For example, CPU utilization can be tracked at the container or virtual machine level through OS functions such as cgroups, and GPU activity metrics such as SM utilization can be obtained through device-level monitoring interfaces (e.g., NVIDIA SMI) with process-level visibility. In contrast to resources such as storage or network subsystems, whose per-process behavior is difficult to isolate due to interference, compute components like CPU and GPU provide relatively stable and attributable utilization signals for practical accounting purposes.

While this study does not propose a specific pricing formula, it establishes empirical foundations for energy-aligned cloud charging mechanisms grounded in workload power profiles. By shifting attention from allocation ratios to workload-driven utilization characteristics, cloud providers may better align pricing with actual operational energy cost drivers in AI-intensive environments.

### 4.3. Limitations of Current Experiment

This study is conducted on an x86 server equipped with a single-socket CPU, a single-DIMM memory configuration, and a GV100-class GPU. Accordingly, the observed allocation insensitivity and utilization-driven power behavior should be interpreted within this specific hardware setting. In particular, large-scale cloud systems may introduce additional factors such as NUMA locality, remote memory access, interconnect overhead, and stronger memory bandwidth contention, which are not captured in this study.

The experimental methodology focuses on isolating intrinsic workload-level power behavior under controlled resource partitioning. To this end, Linux cgroup v2 is used to emulate resource allocation without introducing virtualization layers. As a result, the measured power reflects workload-driven behavior rather than full system-level energy consumption, which would include hypervisor overhead, network and storage I/O, and other platform-level components commonly present in production cloud environments.

In addition, the experiments are conducted in a non-partitioned GPU environment to avoid fine-grained interference effects. Modern GPU sharing mechanisms such as MIG and MPS introduce additional layers of scheduling and contention that depend on GPU architecture and runtime interaction patterns. Consequently, the observed power behavior in such environments may reflect both intrinsic workload characteristics and hardware-dependent sharing dynamics, which are not explicitly considered in this study.

Another limitation arises from the workload selection. In the evaluated workloads, GPU-dominant applications exhibit relatively stable and low CPU utilization, indicating that CPU primarily acts as a supporting pipeline. As a result, no workload simultaneously exhibits high CPU and GPU power. However, in other applications—such as training pipelines with heavy preprocessing [29] or large-scale recommendation systems [30]—both CPU and GPU may contribute substantially to system power, leading to hybrid power behavior not captured in this study. Even in such cases, however, the fundamental principle of energy attribution based on CPU and SM utilization remains applicable within the proposed framework.

Finally, while the analysis suggests that GPU SM utilization dominates dynamic power variation and that memory-related power remains relatively stable across workloads, these observations are derived from a specific GPU architecture. Although memory power is largely associated with static components such as refresh and background circuitry, which vary less with workload behavior, different architectures may exhibit different absolute power characteristics.

## 5. Related Work

Utilization-driven energy accounting emerged from the fundamental limitation that hardware power meters cannot measure the granular energy consumption of individual software entities. Early foundational research established the core principles of this field; for instance, Bellosa [31] introduced event-driven energy accounting by mapping hardware counters to power draw, while Kansal et al. [32] pioneered software-based power models (e.g., Joulemeter) to apportion total system energy among active virtual machines based on CPU and memory utilization.

Building upon these foundations, adapting energy models for cloud multi-tenant scenarios has become a critical research focus. In shared cloud environments, resource contention and virtualization overheads make accurate power attribution significantly more complex. Recent studies have developed fine-grained, isolation-aware accounting methods to measure the energy footprint of individual tenants without requiring physical hardware segregation. For example, frameworks like FaaSMeter [33] have demonstrated that aggregating resource utilization metrics (CPU, memory, and I/O) is an effective proxy for attributing energy costs among concurrent workloads in highly multi-tenant serverless architectures. While such approaches focus on attributing system-level energy to individual tenants, they do not examine how resource allocation itself affects system-level power behavior. In contrast, our work investigates this relationship through direct measurement in co-located AI workloads.

With the rapid proliferation of artificial intelligence, recent research has increasingly focused on characterizing the energy behavior of AI workloads. AI applications, particularly Large Language Models (LLMs) and deep learning inference tasks, exhibit bursty and heterogeneous utilization patterns across accelerators (e.g., GPUs, TPUs), along with high memory bandwidth demands. Prior studies have therefore proposed domain-specific energy models that capture the non-linear relationship between accelerator utilization, model characteristics, and power consumption [34], as well as factors such as GPU idle states and amortized inference costs in distributed environments [35]. However, these approaches primarily rely on utilization- or model-driven abstractions and often assume that energy behavior can be sufficiently explained by resource usage metrics or model parameters alone. In contrast, our work reveals through measurement-driven experiments that deployment configuration (e.g., CPU vs. GPU execution) can fundamentally shift the power regime of the same workload, indicating that power behavior is not solely determined by utilization or model characteristics but also by execution context.

Recent research has increasingly integrated utilization-driven modeling with workload-specific pattern analysis to optimize shared AI clusters. For instance, Kakolyris et al. [36] proposed fine-grained energy models for LLM inference, demonstrating that task-level parameters can non-linearly affect GPU power consumption. Similarly, architectures such as HeShare [37] aim to improve energy efficiency in multi-tenant environments through utilization-aware scheduling and dynamic GPU sharing. While these approaches advance energy-aware optimization in shared systems, they fundamentally rely on utilization-driven abstractions for modeling and decision-making. In contrast, our work provides a complementary perspective by experimentally validating the stability and separability of workload-level power profiles under controlled co-location settings, and further deriving a utilization-driven categorization framework.

## 6. Conclusions

This paper analyzed the energy behavior of widely used AI workloads together with conventional non-AI workloads through a comprehensive experimental study. Our results show that total system power is largely insensitive to resource allocation among co-located workloads, except in cases of insufficient provisioning. At the workload level, power profiles remain stable under heterogeneous co-location and are consistent with standalone executions. Based on measured power decomposition, we derived a workload categorization framework that distinguishes GPU-dominant and CPU-dominant workloads, further characterized along utilization and memory dimensions. From an energy perspective, CPU utilization and SM utilization emerge as the primary determinants of power magnitude in CPU- and GPU-dominant workloads, respectively, while memory-related parameters contribute only marginally to overall power. These findings provide practical implications

for energy-aware cloud management. Although aggregate CPU and GPU power can be measured at the system level, attributing energy consumption to individual workloads remains challenging in multi-tenant environments. Our analysis suggests that workload-level power can be approximated by partitioning measured resource-level power (CPU or GPU) according to corresponding utilization statistics (CPU utilization or SM utilization). This establishes an empirical basis for workload-aware, utilization-driven energy accounting in AI-intensive cloud systems.

While this study provides a controlled analysis of workload-level energy behavior, several directions remain for future work. As our experiments were performed under a controlled single-node system, the findings are limited in scope and should not be overgeneralized to production cloud environments. Extending the analysis to multi-socket systems and heterogeneous architectures (e.g., ARM-based platforms) would enable validation of the proposed observations under more realistic cloud configurations. In addition, incorporating virtualization layers and full system components—including hypervisor overhead, network, and storage subsystems—would allow a more comprehensive evaluation of energy behavior in production cloud environments. Finally, while this study demonstrates that utilization-based metrics provide a practical basis for energy attribution, further validation is necessary to assess the impact of interference, scheduling dynamics, and measurement granularity on attribution accuracy in real deployments.

**Author Contributions:** Conceptualization, K.C. and H.B.; Methodology, K.C.; Software, J.K.; Validation, J.K.; Formal analysis, K.C. and H.B.; Investigation, J.K. and H.B.; Resources, H.B.; Data curation, J.K.; Writing—original draft, J.K. and H.B.; Writing—review and editing, J.K., W.S., S.K., K.C. and H.B.; Visualization, W.S., J.K. and S.K.; Supervision, H.B.; Project administration, H.B.; Funding acquisition, H.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) under Grant RS-2024-00459026 (Energy-Aware Operating System for Disaggregated System) and the National Research Foundation of Korea (NRF) under Grant RS-2024-00461678 funded by the Korean government (MSIT).

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Mashayekhy, L.; Nejad, M.M.; Grosu, D.; Vasilakos, A.V. An online mechanism for resource allocation and pricing in clouds. *IEEE Trans. Comput.* **2016**, *65*, 1172–1184. [CrossRef]
2. Kozhipurath, J. Cloud service costing challenges. In Proceedings of the 2012 IEEE International Conference on Cloud Computing in Emerging Markets, Bangalore, India, 11–12 October 2012; pp. 1–6. [CrossRef]
3. Gnibga, W.; Chien, A.; Blavette, A.; Orgerie, A. FlexCoolDC: Datacenter cooling flexibility for harmonizing water, energy, carbon, and cost trade-offs. In Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems, Singapore, 4–7 June 2024; pp. 108–122. [CrossRef]
4. Sheng, Y.; Zhang, C.; Zhu, Z.; Xu, H.; Wen, J.; Wang, R.; Yang, J.; Wang, Q.; Bu, S. Power for AI data centers: Energy demand, grid impacts, challenges and perspectives. *Energies* **2026**, *19*, 722. [CrossRef]
5. Masanet, E.; Shehabi, A.; Lei, N.; Smith, S.; Koomey, J. Recalibrating global data center energy-use estimates. *Science* **2020**, *367*, 984–986. [CrossRef] [PubMed]
6. Islam, M.; Ren, S. A new perspective on energy accounting in multi-tenant data centers. In Proceedings of the USENIX Workshop on Cool Topics on Sustainable Data Centers (CoolDC), Santa Clara, CA, USA, 19 March 2016.
7. Alnori, A.; Djemame, K.; Alsenani, Y. Agnostic energy consumption models for heterogeneous GPUs in cloud computing. *Appl. Sci.* **2024**, *14*, 2385. [CrossRef]

8. Hu, Q.; Sun, P.; Yan, S.; Wen, Y.; Zhang, T. Characterization and prediction of deep learning workloads in large-scale GPU datacenters. In Proceedings of the ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC), St. Louis, MO, USA, 14–19 November 2021; pp. 1–15. [CrossRef]
9. Lee, J.; Lim, S.; Bahn, H. Write cache management for periodic storage flush in mobile deep learning workloads. In Proceedings of the IEEE International Conference on Advances in Electrical Engineering and Computer Applications, Dalian, China, 22–24 August 2025; pp. 808–812. [CrossRef]
10. Ferroni, M.; Corna, A.; Damiani, A.; Brondolin, R.; Colmenares, J.A.; Hofmeyr, S.; Kubiatowicz, J.D.; Santambrogio, M.D. Power consumption models for multi-tenant server infrastructures. *ACM Trans. Archit. Code Optim.* **2017**, *14*, 38. [CrossRef]
11. Guan, X.; Bashir, N.; Irwin, D.; Shenoy, P. WattScope: Non-intrusive application-level power disaggregation in datacenters. *ACM SIGMETRICS Perform. Eval. Rev.* **2024**, *51*, 24–25. [CrossRef]
12. Mars, J.; Tang, L.; Hundt, R.; Skadron, K.; Soffa, M.L. Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations. In Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44), Porto Alegre, Brazil, 3–7 December 2011; pp. 248–259. [CrossRef]
13. Vamja, T.; Ray, K.; George, F.; Devi, U. On the partitioning of GPU power among multi-instances. *arXiv* **2025**, arXiv:2501.17752. [CrossRef]
14. Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Gener. Comput. Syst.* **2012**, *28*, 755–768. [CrossRef]
15. Kwon, G.; Bahn, H. Energy efficiency analysis of real-time IoT systems under DVFS, offloading, and task load variations. In Proceedings of the IEEE International Conference on Advances in Electrical Engineering and Computer Applications, Dalian, China, 22–24 August 2025; pp. 224–229. [CrossRef]
16. Khan, K.N.; Hirki, M.; Niemi, T.; Nurminen, J.K.; Ou, Z. RAPL in action: Experiences in using RAPL for power measurements. *ACM Trans. Model. Perform. Eval. Comput. Syst.* **2018**, *3*, 1–26. [CrossRef]
17. NVIDIA Corporation. NVIDIA System Management Interface. Available online: https://developer.nvidia.com/system-management-interface (accessed on 22 February 2026).
18. Linux Kernel Organization. Control Groups v2 (cgroup v2). Linux Kernel Documentation. 2024. Available online: https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v2.html (accessed on 22 February 2026).
19. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An imperative style, high-performance deep learning library. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS), Vancouver, BC, Canada, 8–14 December 2019; pp. 8024–8035.
20. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
21. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Master's Thesis, University of Toronto, Toronto, ON, Canada, 2009.
22. Jocher, G.; Chaurasia, A.; Qiu, J. Ultralytics YOLOv8. 2023. Available online: https://github.com/ultralytics/ultralytics (accessed on 22 February 2026).
23. Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X.V.; et al. OPT: Open pre-trained transformer language models. *arXiv* **2022**, arXiv:2205.01068. Available online: https://arxiv.org/abs/2205.01068 (accessed on 22 February 2026).
24. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models Are Unsupervised Multitask Learners. OpenAI Technical Report. 2019. Available online: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (accessed on 22 February 2026).
25. FFmpeg Developers. FFmpeg: A Complete, Cross-Platform Solution to Record, Convert and Stream Audio and Video. 2024. Available online: https://ffmpeg.org (accessed on 22 February 2026).
26. OpenJS Foundation. Node.js. 2024. Available online: https://nodejs.org (accessed on 22 February 2026).
27. Vogelsang, T. Understanding the energy consumption of dynamic random access memories. In Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO), Atlanta, GA, USA, 4–8 December 2010; pp. 363–374. [CrossRef]
28. Mohan, J.; Phanishayee, A.; Raniwala, A.; Chidambaram, V. Analyzing and mitigating data stalls in DNN training. *Proc. VLDB Endow.* **2021**, *14*, 771–784. [CrossRef]
29. Huang, Y.; Cheng, Y.; Bapna, A.; Firat, O.; Chen, D.; Chen, M.; Lee, H.; Ngiam, J.; Le, Q.V.; Wu, Y. GPipe: Efficient training of giant neural networks using pipeline parallelism. *Adv. Neural Inf. Process. Syst.* **2019**, *32*. [CrossRef]
30. Naumov, M.; Mudigere, D.; Shi, H.-J.M.; Huang, J.; Sundaraman, N.; Park, J.; Wang, X.; Gupta, U.; Wu, C.-J.; Azzolini, A.G.; et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv* **2019**, arXiv:1906.00091. [CrossRef]

31. Bellosa, F. The benefits of event-driven energy accounting in power-sensitive systems. In Proceedings of the 9th ACM SIGOPS European Workshop: Beyond the PC: New Challenges for the Operating System, Kolding, Denmark, 17–20 September 2000; pp. 37–42. [CrossRef]
32. Kansal, A.; Zhao, F.; Liu, J.; Kothari, N.; Bhattacharya, A. Virtual machine power metering and provisioning. In Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC), Indianapolis, IN, USA, 10–11 June 2010; pp. 39–50. [CrossRef]
33. Sharma, P.; Fuerst, A. Accountable carbon footprints and energy profiling for serverless functions. In Proceedings of the ACM Symposium on Cloud Computing (SoCC), Redmond, WA, USA, 20–22 November 2024; pp. 522–541. [CrossRef]
34. Kasioulis, M.; Symeonides, M.; Ioannou, G.; Pallis, G.; Dikaiakos, M.D. Energy modeling of inference workloads with AI accelerators at the edge: A benchmarking study. In Proceedings of the IEEE International Conference on Cloud Engineering (IC2E), Paphos, Cyprus, 24–27 September 2024. [CrossRef]
35. Trihinas, D.; Thamsen, L.; Beilharz, J.; Symeonides, M. Towards energy consumption and carbon footprint testing for AI-driven IoT services. In Proceedings of the IEEE International Conference on Cloud Engineering (IC2E), Pacific Grove, CA, USA, 26–30 September 2022. [CrossRef]
36. Wilkins, G.; Keshav, S.; Mortier, R. Offline energy-optimal LLM serving: Workload-based energy models for LLM inference on heterogeneous systems. *ACM SIGENERGY Energy Inform. Rev.* **2024**, *4*, 113–119. [CrossRef]
37. Jiang, Z.; Cai, Z.; Zhao, H.; Zhang, B.; Wu, T.; Qiang, Y. HeShare: Energy-aware and efficient multi-task GPU sharing in heterogeneous GPU-based computing systems. *IEEE Trans. Comput.* **2025**, *75*, 776–787. [CrossRef]


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.