---

title: "Benchmarking GenAI on End-User Devices with ConsumerBench" date: "October 14, 2025" author: "Yile Gu, Rohan Kadekodi, Hoang Nguyen, Keisuke Kamahori, Yiyu Liu, Baris Kasikci" authorLinks:

- name: "Yile Gu" url: "https://ikace.github.io/"
- name: "Rohan Kadekodi" url: "https://homes.cs.washington.edu/~rohankad/"
- name: “Hoang Nguyen” url: “https://www.kipiiler.me/”
- name: “Keisuke Kamahori” url: “https://kamahori.org/”
- name: “Yiyu Liu” url: “https://lau.yeeyu.org/”
- name: "Baris Kasikci" url: "http://bariskasikci.org/"
  excerpt: "We present ConsumerBench, the first comprehensive benchmarking framework for generative-AI applications running directly on end-user devices"

TL;DR: We present **ConsumerBench**, the first comprehensive benchmarking framework for generative AI applications running directly on end-user devices. Unlike traditional benchmarks that assume exclusive GPU access in cloud datacenters, ConsumerBench captures the realistic, concurrent execution of multiple GenAI apps such as chatbots, image generation, live captioning, research agents, etc., competing for limited local resources. It measures both application-level SLOs (latency, throughput) and system-level metrics (GPU utilization, memory bandwidth, power) to reveal critical inefficiencies and guide the design of efficient on-device AI systems.

Paper: https://arxiv.org/abs/2506.17538

Code: https://github.com/efeslab/ConsumerBench

Benchmarking On-Device AI

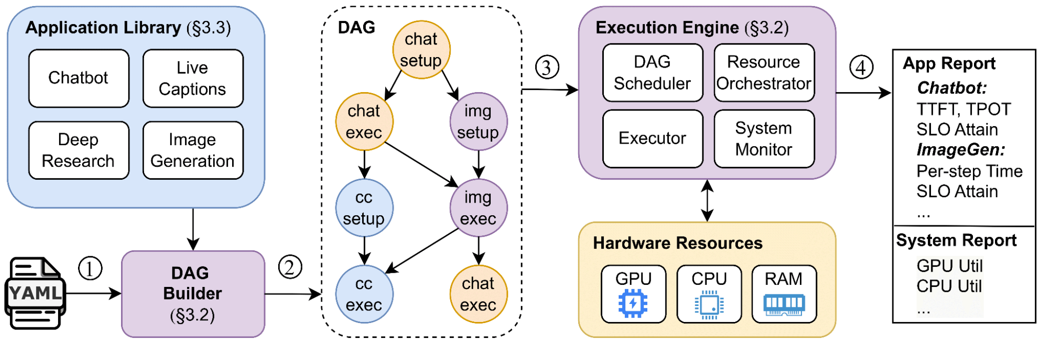
Generative AI has shifted from cloud servers to **local devices** such as laptops and smartphones (https://arxiv.org/pdf/2506.02153). This migration is motivated by privacy, low latency, and offline availability—but it introduces new systems challenges.

Unlike cloud environments, **end-user devices must juggle multiple concurrent applications**, each powered by distinct AI models and each with different service-level objectives (SLOs). A live captioning app demands millisecond-level latency, while a deep research agent can run in the background for minutes. When they share a single GPU, contention and unfair scheduling can easily degrade user experience.

Existing benchmarks, however, evaluate models in isolation. They measure throughput and energy efficiency on dedicated hardware but overlook what happens when models compete. ConsumerBench fills this gap by modeling real multi-application execution on resource-constrained hardware.

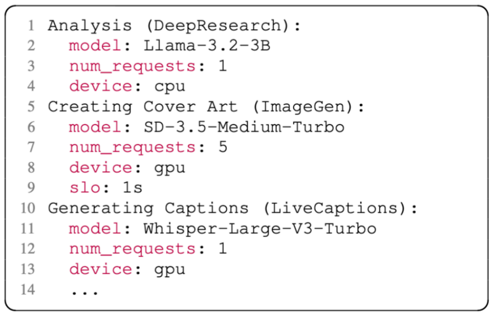
ConsumerBench

ConsumerBench is a benchmarking framework that lets researchers and developers evaluate how GenAI applications behave under realistic, concurrent workloads on end-user hardware. Given a user configuration of the workflow, ConsumerBench orchestrates the execution through a graph representation. After the workflow finishes, it generates a benchmark report for SLO satisfaction and resource efficiency.



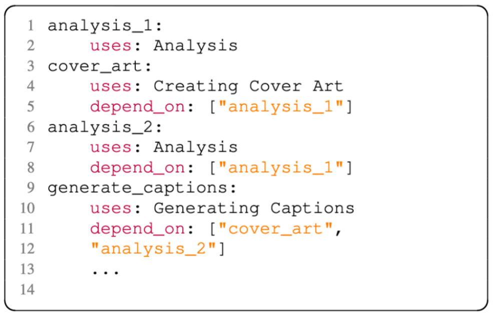
Key Features

- **Multi-application workflows:** Define tasks and their dependencies (e.g., Chatbot → ImageGen → LiveCaptions) using simple YAML configuration.
- **SLO tracking:** Measure per-app latency and attainment of user-defined SLOs.
- **System-level monitoring:** Collect GPU/CPU utilization, memory bandwidth, and power metrics through NVIDIA DCGM and Intel PCM.
- **Flexible resource orchestration:** Evaluate greedy GPU allocation, static partitioning, or shared inference servers.
- **Extensibility:** Plug in custom GenAI apps by implementing a standard `setup()`, `execute()`, and `cleanup()` interface.



```
Analysis (DeepResearch):
  model: Llama-3.2-3B
  num_requests: 1
  device: cpu
Creating Cover Art (ImageGen):
  model: SD-3.5-Medium-Turbo
  num_requests: 5
  device: gpu
  slo: 1s
Generating Captions (LiveCaptions):
  model: Whisper-Large-V3-Turbo
  num_requests: 1
  device: gpu
  ...
```

(a) Task Definition



```
analysis_1:
    uses: Analysis
cover_art:
    uses: Creating Cover Art
    depend_on: ["analysis_1"]
analysis_2:
    uses: Analysis
    depend_on: ["analysis_1"]
generate_captions:
    uses: Generating Captions
    depend_on: ["cover_art",
    "analysis_2"]
    ...
```

(b) Workflow Definition

Architecture Overview

1. Users define applications and SLOs in YAML
2. ConsumerBench builds a **DAG workflow** connecting apps via dependencies.
3. A **scheduler** and **resource orchestrator** coordinates execution and collects metrics.
4. After completion, ConsumerBench generates detailed reports summarizing SLO attainment and resource efficiency.

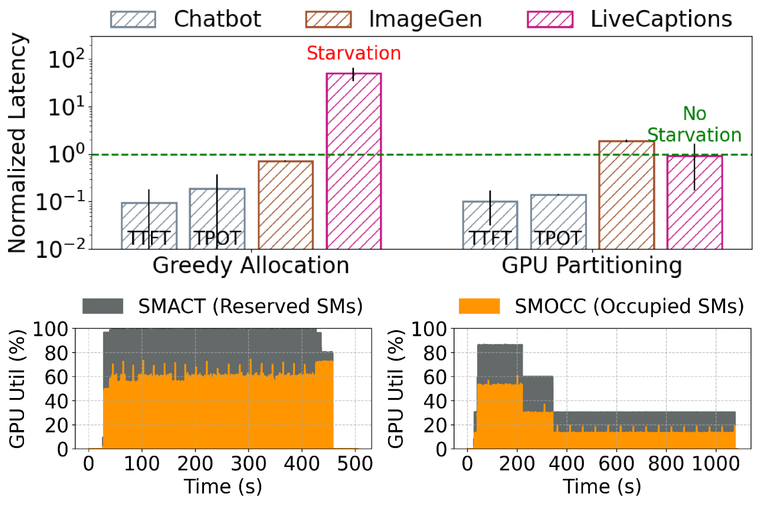
Currently Supported Applications

Application	**Modality**	**Model**	**Example SLOs**
Chatbot	text → text	Llama-3.2-3B	TTFT 1 s, TPOT 0.25 s
DeepResearch	multi-step agent	Llama-3.2-3B	N/A
ImageGen	text → image	SD-3.5-Medium-Turbo	1s per denoising step
LiveCaptions	audio → text	Whisper-Large-v3-Turbo	2s per audio chunk

These applications span interactive and background workloads, reflecting the diversity of modern consumer AI use cases.

Key Evaluation Results

- Benchmarked on a **consumer-grade RTX 6000 GPU** and a **MacBook M1 Pro** laptop.
- Tracks GPU utilization (SMACT vs SMOCC), power, and memory bandwidth in real time.
- Demonstrates performance disparities between architectures such as NVIDIA GPUs and Apple Silicon



Resource Contention Hurts Fairness

Under **greedy GPU allocation**, large-kernel workloads like image generation monopolize GPU resources, starving lighter tasks like live captioning—causing up to **12.4× latency increase** and near-zero SLO attainment.

Static Partitioning Wastes Performance

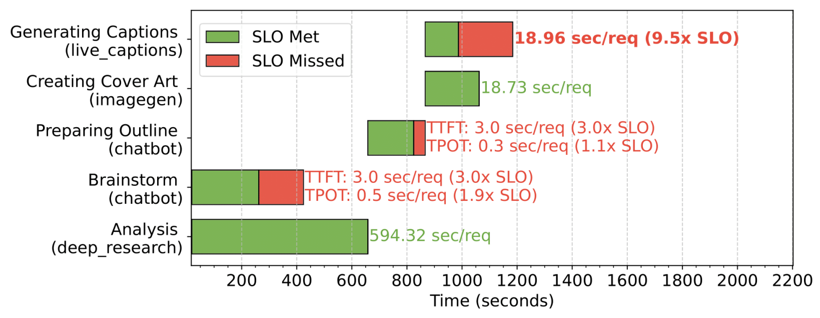
Evenly dividing GPU resources (e.g., via NVIDIA MPS) improves fairness but leads to **under-utilization**: idle partitions cannot borrow unused compute, lowering throughput despite available capacity.

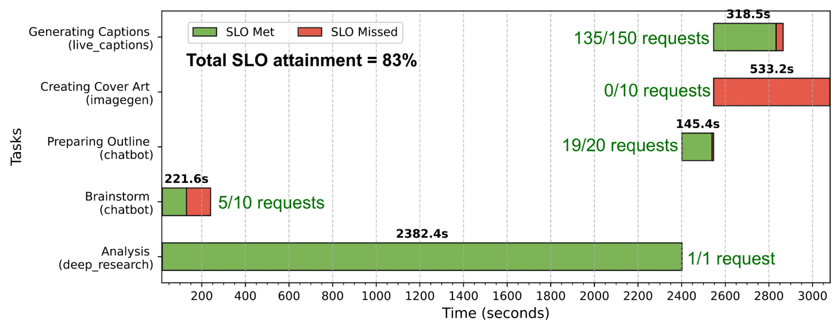
Need for dynamic GPU scheduling Policies are Insufficient

There is a need for dynamic GPU scheduling policies that are SLO-aware and that transparently and dynamically share a single GPU's resources across multiple GenAI applications. Existing dynamic scheduling systems (Orion, REEF, etc) fail to run applications off-the-shelf without modifications or manual inspection of the CUDA kernels.

Inference Servers Aren't One-Size-Fits-All

When applications share a single model through an inference server (e.g., Chatbot + DeepResearch on llama.cpp), fixed configurations like large KV caches can **harm interactive latency**, revealing the need for **SLO-aware adaptive servers**.





Real-World Workflows Expose Trade-offs

In a realistic content-creation workflow (brainstorm → analysis → cover art → captioning), greedy scheduling cuts total runtime by 45% but risks starvation; partitioning improves fairness but increases completion time. The result: **no single policy fits all scenarios—**systems must adapt dynamically.

Insights for Future Design

- **Architectural Efficiency:** Design model kernels to maximize SM occupancy and minimize register pressure.
- **Concurrency-Aware Kernels:** Anticipate multi-app co-execution when implementing GPU kernels.
- **Flexible Resource Management:** Move beyond static GPU partitions toward **dynamic, SLO-aware scheduling** that balances fairness and efficiency.

Why It Matters

ConsumerBench bridges a major gap between **cloud-centric AI benchmarking** and **real-world on-device deployment**. It enables:

- **Model developers** pinpoint inefficiencies in their kernels.
- **System designers** explore new scheduling and memory-management strategies.
- **Device manufacturers** understand how different workloads coexist on shared hardware.

By treating the *end-user device* as a first-class platform for AI, ConsumerBench paves the way toward responsive, energy-efficient, and fair on-device intelligence.

Contributions

Please check out the code and feel free to post any requests or bug reports at our GitHub Issues.

We welcome and encourage contributions to the ConsumerBench repository, especially for:

- Adding new applications
- Adding new realistic user workflows
- Evaluating with different GPUs and GPU scheduling mechanisms