

Deep-Learning Inference Performance Across Heterogeneous HPC CPU Systems

Calebe P. Bianchini^{1,3}, Evaldo Costa¹, Felipe Portella²,
Gabriel Freytag¹, Jeferson S. Rocha², Leon Barboza¹,
Maicon M. Alves², Rodrigo Machado¹, Vinícius Ventura¹

¹ CESAR Innovation Center, Recife, PE, Brazil

²Petróleo Brasileiro S.A. (PETROBRAS), Rio de Janeiro, RJ, Brazil

³Mackenzie Presbyterian University, São Paulo, SP, Brazil

{cpb,ebc,gf2,lmub,rcm,vlv}@cesar.org.br

{felipeportella,jsrocha,maicon.melo}@petrobras.com.br

calebe.bianchini@mackenzie.br

Abstract. *Choosing a CPU platform for deep-learning inference in heterogeneous HPC systems is not straightforward because performance changes with the workload and execution configuration. We compare five HPC CPU configurations using Detectron2 and MLPerf Inference v5.1 Offline, complemented by controlled BERT and ResNet50 scaling and repeated-run experiments. No platform leads all 11 workloads; median rank similarity is 0.700 within Detectron2, 0.400 within MLPerf, and -0.100 across suites. Scaling is sublinear at every tested multiplier, while cross-platform CV reaches 16.0%. Overall, the results favor reporting performance together with its workload, allocation, placement, and measurement context rather than as a single global platform ranking.*

1. Introduction

Deep-learning inference on CPUs spans a wide range of systems, from edge processors to large server and HPC nodes. In heterogeneous HPC platforms, CPUs differ substantially in ISA, core count, memory topology, and software stack, and often coexist with accelerators. These differences make platform selection less straightforward than a single throughput number suggests: a system that performs well for one workload may not retain the same position after changing the model, benchmark family, resource allocation, or placement. Modern heterogeneous HPC systems also expose substantial CPU resources alongside accelerators, making CPU-side inference performance relevant to workload placement and system-level characterization. Prior studies already show that CPU inference is strongly shaped by model structure and implementation choices [Mittal et al. 2022, Liu et al. 2019].

Previous work has examined these effects from complementary angles. Cross-architecture studies already compare inference on Intel, AMD, and Arm CPUs [Liu et al. 2019, Ahn et al. 2023], while characterization and optimization work shows that models, frameworks, threading, affinity, and NUMA placement can reshape achieved performance [Prieto et al. 2023, Pati et al. 2022,

Jiang et al. 2023, Camargo et al. 2021, Denoyelle et al. 2019]. MLPerf Inference provides a common methodology for deployment-oriented measurements [Reddi et al. 2020, Reddi et al. 2021], and systems-benchmarking research emphasizes repeated measurements and explicit treatment of variability [Hoeffler and Belli 2015, Kalibera and Jones 2013, Chunduri et al. 2017]. Our focus is narrower than establishing another x86-versus-Arm comparison: we examine deployed, high-core-count HPC CPU configurations and ask whether their relative ordering persists across workloads and benchmark families while scaling, placement, and run stability are analyzed separately.

We study five measured HPC CPU configurations: AMD EPYC, NVIDIA Grace, the CPU side of GH200, the host CPU of an H100 system, and the CPU side of MI300A. Cross-platform experiments use two independent benchmark families under a uniform methodology within each family: seven Detectron2 detection/segmentation configurations and four MLPerf Inference v5.1 Offline workloads (BERT, ResNet50, RetinaNet, and 3D-UNet). A separate controlled campaign evaluates CPU scaling and compact-versus-spread placement for BERT and ResNet50 on AMD EPYC, Grace, and GH200 CPU. Accelerator execution is excluded. Because the systems expose unequal CPU allocations, the cross-platform results describe the measured configurations rather than an equal-core architecture comparison.

Our analysis combines four views of the same problem: delivered cross-platform throughput, consistency of platform rankings across workloads, controlled resource-scaling and placement experiments, and run-to-run stability with follow-up campaigns kept separate from the predefined measurements.

The paper makes three contributions:

- Across 11 workloads on five systems, we quantify how much platform orderings persist within and across two benchmark suites instead of reducing the comparison to one aggregate winner.
- Controlled BERT and ResNet50 experiments on AMD EPYC, Grace, and GH200 CPU expose sublinear scaling and placement-dependent behavior; an independent campaign reproduces the direction of the GH200 BERT throughput drop from 72 to 144 compact CPUs.
- Eleven valid MLPerf executions per cross-platform cell allow us to report run-to-run variability separately from per-run benchmark validity and from follow-up diagnostic observations.

The results show why these distinctions matter. AMD EPYC takes first place in six of the 11 system-throughput comparisons, Grace in three, and GH200 CPU in two. Median Spearman rank similarity is 0.700 within Detectron2 and 0.400 within MLPerf, but only -0.100 across suites. Every reported $2\times$ and $4\times$ scaling point is sublinear, placement effects change with both platform and workload, and some valid MLPerf cells show substantial run-to-run dispersion. A single platform score would hide much of this behavior.

The remainder of the paper is organized as follows. Section 2 introduces the benchmark concepts and related work. Section 3 describes the experimental methodology, Section 4 presents the results, and Section 5 discusses their implications and limitations. Section 6 concludes the paper.

2. Background and Related Work

2.1. Benchmark and metric concepts

MLPerf Inference defines common models, scenarios, metrics, accuracy targets, and a load generator for comparable inference measurements [Reddi et al. 2020, Reddi et al. 2021]. In the Offline scenario, throughput is measured for a stream of samples without a per-query latency constraint. Our campaign follows the v5.1 reference configuration, whose official files specify a 600 s minimum Offline duration, workload-specific minimum query counts, and one normative performance run per workload [MLCommons 2025]. A run that satisfies these rules is valid under the benchmark contract; the rules do not describe how much another independently valid run may differ. We analyze run-to-run stability separately. Related work has also examined how MLPerf results should be interpreted across configurations and deployment environments [Hodak et al. 2022, Elordi et al. 2021].

Detectron2 forms a separate experiment family because its timed region and workload semantics differ from MLPerf. Our primary Detectron2 metric is steady-state model-forward throughput over preprocessed inputs at CPU batch size 1. Since the two suites time different regions and report different throughput units, we do not compare their absolute values directly. Cross-suite analysis uses only the relative ordering of platforms within each workload.

Throughout the paper, T denotes delivered throughput and N the allocated CPU count. We report T/N as CPU-resource-normalized throughput, and use $S_k = T(kN)/T(N)$ with $E_k = S_k/k$ for controlled within-platform scaling. Fixed-resource topology effects compare spread and compact CPU placements, run-to-run dispersion is summarized with coefficient of variation (CV), and platform-order similarity with descriptive Spearman rank correlation. Table 3 gives the interpretation used for each metric; no global score or formal performance-portability metric is computed.

2.2. Cross-platform CPU inference

Prior CPU-inference work spans both cross-architecture comparison and architecture-specific characterization. Liu et al. [Liu et al. 2019] evaluate 15 CNNs on Intel Skylake, AMD EPYC, and Arm Cortex-A72 CPUs while varying thread count and applying full-stack optimizations. Ahn et al. [Ahn et al. 2023] use MLPerf Edge, including Offline measurements, to compare x86 servers with a Raspberry Pi Arm platform. Complementary studies show how model structure, operators, libraries, and processor organization affect CPU behavior [Prieto et al. 2023, Mittal et al. 2022].

Arm inference has likewise been studied directly. Castelló et al. [Castelló et al. 2022] optimize ResNet50 on an Arm Carmel CPU, and Jiang et al. [Jiang et al. 2023] study transformer inference with NUMA-aware scheduling on Arm many-core CPUs. Together with the cross-architecture studies above, these results establish strong workload/runtime sensitivity. Our study builds on that foundation by asking whether the *relative ordering* of HPC CPU platforms persists across workloads and across two benchmark families.

Benchmark methodology provides a related caution against over-generalizing isolated results. Alvarado Rodriguez et al. [Alvarado Rodriguez et al. 2020] compare

inference benchmarking approaches, while Hodak et al. [Hodak et al. 2022] show that conclusions can change with workload and scenario. This motivates analyzing ranking consistency directly rather than constructing an aggregate winner.

2.3. CPU scaling, affinity, and topology

Adding CPU resources does not guarantee monotonic inference gains. Liu et al. [Liu et al. 2019] report that additional OpenMP threads can stop helping or reduce throughput, and Pati et al. [Pati et al. 2022] show that BERT places different demands on the system across operators and input characteristics. Runtime configuration is another source of variation: Mebratu et al. [Mebratu et al. 2021] tune TensorFlow CPU-backend parameters, Camargo et al. [Camargo et al. 2021] measure thread-mapping effects, and Khan and Islam [Khan and Islam 2026] study CPU scheduling and thread affinity on multi-core laptops. REDUCT [Nori et al. 2021] analyzes scaling bottlenecks, VELTAIR studies multi-tenant scheduling and resource interference [Liu et al. 2022], and Vellaisamy et al. [Vellaisamy et al. 2025] use fine-grained profiling to expose CPU-side bottlenecks on coupled CPU–GPU systems, including GH200. These studies reinforce that nominal core count is an incomplete predictor of throughput and motivate profiling when causal explanations are required.

Placement has long been treated as a first-class concern in HPC. Mazouz et al. [Mazouz et al. 2011] compare thread-pinning strategies on multicore/NUMA systems; Denoyelle et al. [Denoyelle et al. 2019] jointly study compact/scatter thread placement with first-touch/interleave data placement; and Dokulil and Benkner [Dokulil and Benkner 2020] emphasize NUMA-aware core allocation for co-operating applications. These studies motivate explicit control of CPU placement, but they are not end-to-end DNN inference evaluations on the HPC CPUs considered here. We separate resource scaling from fixed-resource compact-versus-spread placement and avoid reading the latter as a direct measurement of NUMA memory placement, which we do not explicitly bind.

2.4. Variability and reproducible benchmarking

Run-to-run variation is a well-established systems concern. Chunduri et al. [Chunduri et al. 2017] characterize variability on Xeon Phi-based Cray XC systems, and Cui et al. [Cui et al. 2023] examine OpenMP variability as thread pinning, SMT, and frequency change. More broadly, Hoefer and Belli [Hoefer and Belli 2015] argue for explicit reporting of nondeterminism and dispersion, while Kalibera and Jones [Kalibera and Jones 2013] show how repetitions can be allocated across experimental levels to obtain interpretable comparisons without unnecessary cost. Deep500 [Ben-Nun et al. 2019] brings similar reproducibility concerns to deep-learning benchmarking through standardized, modular measurement infrastructure.

Our repeated MLPerf runs serve this complementary purpose. MLPerf v5.1 establishes whether each individual run satisfies its validity requirements [MLCommons 2025]; repeated executions let us observe how throughput varies across valid runs. We report CV for the cross-platform cells and keep predefined, replication, and diagnostic campaigns as separate samples. That separation becomes particularly important for the GH200 BERT scaling anomaly discussed later, where follow-up measurements were collected for a different experimental purpose than the primary campaign.

Table 1. CPU systems used in the cross-platform experiments. CPU counts are the allocations used by the benchmarks.

System	ISA	Processor	CPU's	Sockets	Threads/core	CPU-bearing NUMA nodes
AMD EPYC	x86_64	AMD EPYC 9684X	192	2	1	24
Grace	aarch64	Arm Neoverse-V2	144	2	1	2
GH200 CPU	aarch64	Arm Neoverse-V2	288	4	1	4
H100 host CPU	x86_64	Intel Xeon Max 9468	96	2	1	8
MI300A CPU	x86_64	AMD Instinct MI300A	48	2	1	2

Table 2. Experimental design. Performance and accuracy/correctness campaigns are analyzed separately.

Study	Workloads	Primary metric / runtime	Runs / CPU policy
Detectron2	7 detection/segmentation configurations	steady-state model-forward images/s; PyTorch/Detectron2	10 per cell; full-node
MLPerf Offline	BERT, ResNet50, RetinaNet, 3D-UNet	samples/s; ONNX Runtime 1.23.2	11 valid per cell; full-node
Controlled	BERT, ResNet50	MLPerf/ONNX Runtime throughput	3 per primary configuration; explicit CPU sets

3. Experimental Methodology

The evaluation combines two cross-platform campaigns with one controlled intra-platform campaign. Table 1 lists the five measured CPU configurations and Table 2 summarizes workloads, metrics, and repetition counts. CPU allocations differ across systems, so cross-platform results describe the deployed configurations rather than equal-core architectural performance. Throughout, “CPU” denotes an allocatable logical processing unit exposed to the benchmark by the operating system or scheduler; Table 1 reports one hardware thread per core for every measured configuration. Performance and accuracy/correctness campaigns are analyzed separately.

3.1. Cross-platform campaigns

Detectron2. We evaluate seven configurations: Faster R-CNN R50-C4 and R50-FPN, RetinaNet R50-FPN, Mask R-CNN R50-C4 and R50-FPN on COCO 2017 validation data, Mask R-CNN R50-FPN on LVIS v0.5 validation, and Faster R-CNN R50-C4 on Pascal VOC 2007 test. Each of the 35 system/workload cells has 10 independent performance repetitions, for 350 observations, at CPU batch size 1. The primary metric times repeated model-forward calls over the same preprocessed batch; image decoding and preprocessing are outside the timed region. This design measures steady-state model-forward throughput and can benefit from cache reuse, so it should not be read as an end-to-end new-image ingestion rate. A secondary sensitivity metric times the `DefaultPredictor` path, including preprocessing and postprocessing but excluding disk I/O, solely to test whether the identity of the winning system depends on the primary timing definition.

The Detectron2 environment uses PyTorch 2.9.1, torchvision 0.24.1, NumPy 1.26.4, and Detectron2 commit `02b5c4e295e990042a714712c21dc79b731e8833`. CPU affinity is configured with `OMP_PROC_BIND=close` and `OMP_PLACES=cores`. Where Slurm is used, the jobs request exclusive single-node allocations so that the benchmark sees the intended CPU set. Accuracy runs are separate correctness checks and do not enter the throughput statistics.

Table 3. Metric interpretation used throughout the analysis.

Metric	Definition	Measures	Does not measure
System throughput	T	Delivered inference throughput of the measured system configuration	Architectural/energy efficiency or intrinsic core performance
Throughput per allocated CPU	T/N	CPU-resource productivity of the measured configuration	Equal-core or intrinsic per-core performance
Relative score	$T/\max_j T_j$	Position relative to the best measured system for the same workload	Percentage of theoretical peak or “100% efficiency”
Speedup	S_k	Intra-platform gain from controlled resource scaling	Absolute cross-platform performance
Scaling efficiency	$E_k = S_k/k$	Proximity to ideal linear scaling within a platform/workload	Energy or architectural efficiency
CV	$100s/\bar{T}$	Relative run-to-run variability	Statistical confidence or efficiency
Accuracy	benchmark-specific	Correctness guard	Performance

MLPerf Inference. BERT, ResNet50, RetinaNet, and 3D-UNet use the MLPerf Inference v5.1 reference workflow in the Offline scenario with ONNX Runtime 1.23.2. In v5.1, BERT, ResNet50, and 3D-UNet are Edge-category benchmarks, while RetinaNet supports both Edge and Datacenter; our campaign uses Edge for the first three and Datacenter for RetinaNet [MLCommons 2025]. Each of the 20 system/workload cells contains 11 valid `PerformanceOnly` repetitions, for 220 observations. The workflow-recorded CPU thread count matches the full allocation listed in Table 1 for every retained cross-platform cell. All analyzed runs satisfy the v5.1 minimum-duration and minimum-query-count gates. Find-performance executions serve only for calibration and are excluded from the statistical sample. The official v5.1 configuration files are the reference for the scenario rules used here [MLCommons 2025].

Medians are the primary throughput summaries; quartiles, standard deviation, and CV describe dispersion.

3.2. Metrics and cross-suite comparison

For workload w and system s , relative throughput is $T_{s,w}/\max_j T_{j,w}$. The value is workload-specific and does not represent a fraction of theoretical peak. Throughput per allocated CPU, $T_{s,w}/N_s$, summarizes CPU-resource productivity for the measured allocation.

Because workloads from the two suites have different throughput semantics and scales, systems are ranked independently within each workload. We then compute descriptive Spearman correlation for every pair of rankings. Detectron2–Detectron2, MLPerf–MLPerf, and cross-suite groups contain 21, 6, and 28 workload pairs, respectively; these counts refer to workload comparisons, not experimental repetitions.

3.3. Controlled scaling and topology campaign

The controlled campaign runs BERT and ResNet50 on AMD EPYC, Grace, and GH200 CPU through the same MLPerf/ONNX Runtime execution path. Each primary configuration has three repetitions, so we treat the controlled comparisons descriptively rather than as formal hypothesis tests, consistent with systems-benchmarking guidance on nondeterminism and uncertainty [Hoefer and Belli 2015, Kalibera and Jones 2013]. CPU sets are enforced with Slurm/cgroups, ONNX Runtime intra-op threads are set to the allocated CPU count, and monitoring found no CPU-affinity violations in the retained runs.

Compact allocations use 48, 96, and 192 CPUs on AMD EPYC; 36, 72, and 144 on Grace; and 72, 144, and 288 on GH200 CPU. At fixed resource counts, spread placements distribute CPUs across sockets or CPU-bearing NUMA domains: AMD and Grace include two-domain spread cases, while GH200 includes both two- and four-domain 72-CPU layouts. We do not configure an explicit memory-binding policy. The fixed-resource comparisons capture the placement behavior of the executed configurations but do not establish where model data resided in memory at runtime.

Scaling is reported as $S_k = T(kN)/T(N)$ and $E_k = S_k/k$, relative to the smallest compact configuration on each platform. Topology effects use the throughput ratio or percentage change between spread and compact placements at the same CPU count. The primary GH200 BERT campaign exposed non-monotonic 72→144 scaling and high variability at 288 CPUs, so an independent campaign repeated the relevant configurations and a separate diagnostic campaign added observations at the 288-CPU point. We analyze these three samples separately throughout; follow-up observations are not merged with the predefined primary campaign.

4. Results

4.1. Cross-platform performance

Figure 1 summarizes the five-system comparison. No system leads every workload. AMD EPYC takes first place in six of the 11 system-throughput comparisons, Grace in three, and GH200 CPU in two. Within MLPerf, GH200 CPU leads 3D-UNet and BERT, whereas AMD EPYC leads ResNet50 and RetinaNet; the Detectron2 winners are split between AMD EPYC and Grace. This split shows that the ordering changes with the workload instead of collapsing to one universal winner.

The resource-normalized view changes the ordering more sharply. MI300A CPU ranks first on throughput per allocated CPU for all 11 workloads even though it does not dominate absolute system throughput. With only 48 CPUs allocated, this result describes the productivity of that measured configuration (Table 3); it is not an equal-core comparison. The contrast between the two views shows why CPU allocation must accompany a cross-platform throughput result.

4.2. Workload and suite sensitivity

Platform rankings are more similar within each suite than across suites (Figure 2). Median Spearman rank similarity is 0.700 across the 21 Detectron2 workload pairs and 0.400 across the six MLPerf pairs. Across the 28 cross-suite pairs, the median is -0.100 , indicating little agreement between the two benchmark families in this five-system evidence set. These correlations are descriptive and are not accompanied by inferential tests.

A predefined Detectron2 sensitivity check compares the primary steady-state model-forward metric with the alternative `DefaultPredictor`-based metric. The winning system is unchanged in all 14 checks: seven for absolute throughput and seven for throughput per allocated CPU. The Detectron2 winner labels used in the rank analysis are unchanged under this alternate timing definition, although the two metrics retain different semantics and are not numerically interchangeable.

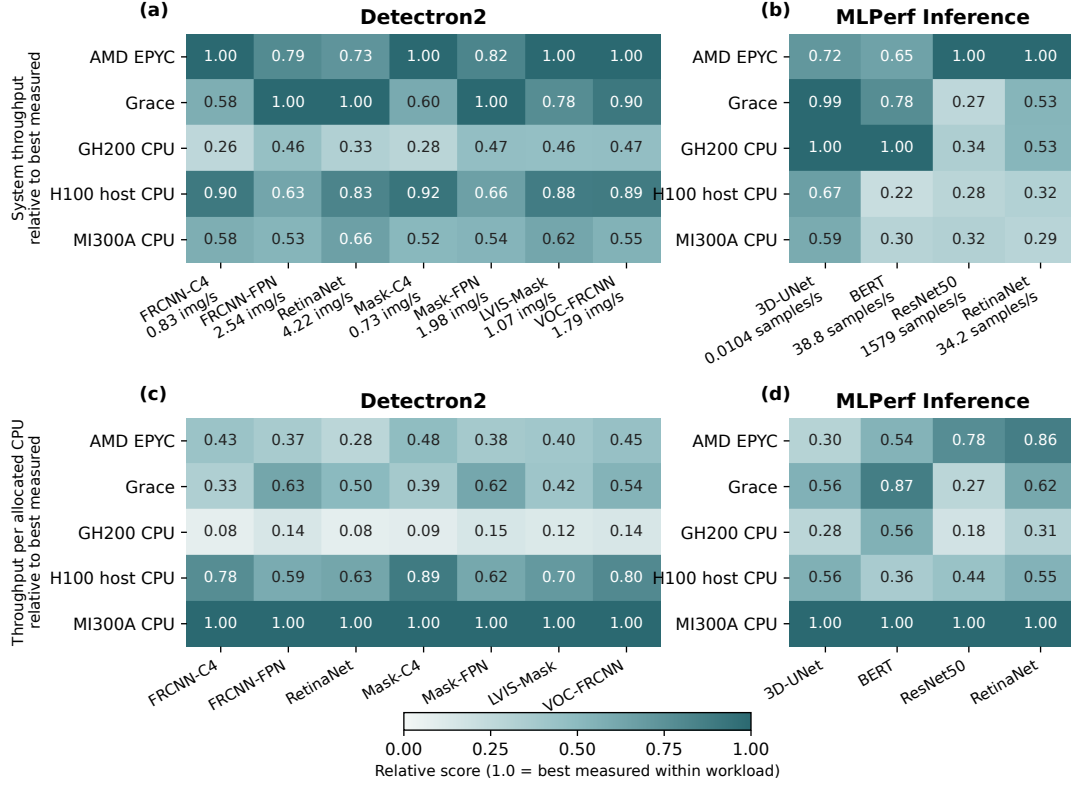


Figure 1. Relative system throughput (top) and throughput per allocated CPU (bottom) for five systems. Scores are normalized to the best measured system within each workload; the top-panel workload labels also report the best absolute median throughput. Absolute values are not compared across suites because their timed regions and throughput semantics differ.

4.3. CPU scaling and topology placement

Figure 3 summarizes the controlled study. Every reported $2\times$ and $4\times$ resource point is sublinear. The shape of the scaling curve depends strongly on the platform and workload: AMD EPYC scales more effectively for ResNet50 than for BERT, Grace shows the opposite tendency at the full-node point, and GH200 BERT is non-monotonic. The clearest case is GH200 BERT at $72 \rightarrow 144$ compact CPUs, where $S_2 = 0.675$ and $E_2 = 0.338$; adding CPUs reduces throughput.

The independent GH200 campaign reproduces the direction of this drop. BERT throughput changes from 17.5720 to 11.1251 samples/s at $72 \rightarrow 144$ CPUs, compared with 17.5995 to 11.8802 samples/s in the primary campaign. The effect size differs, so we describe the result as directional replication rather than identical reproduction.

Placement at fixed CPU count also changes throughput. Spread placement improves both AMD workloads and produces opposite-sign effects for BERT and ResNet50 on Grace. GH200 shows an even sharper workload reversal: among the 72-CPU BERT placements, spread4 is slightly faster than compact while spread2 is slower; for ResNet50, both spread layouts lose throughput and spread4 is the slowest of the three. No placement policy is uniformly favorable across the measured platform/workload combinations.

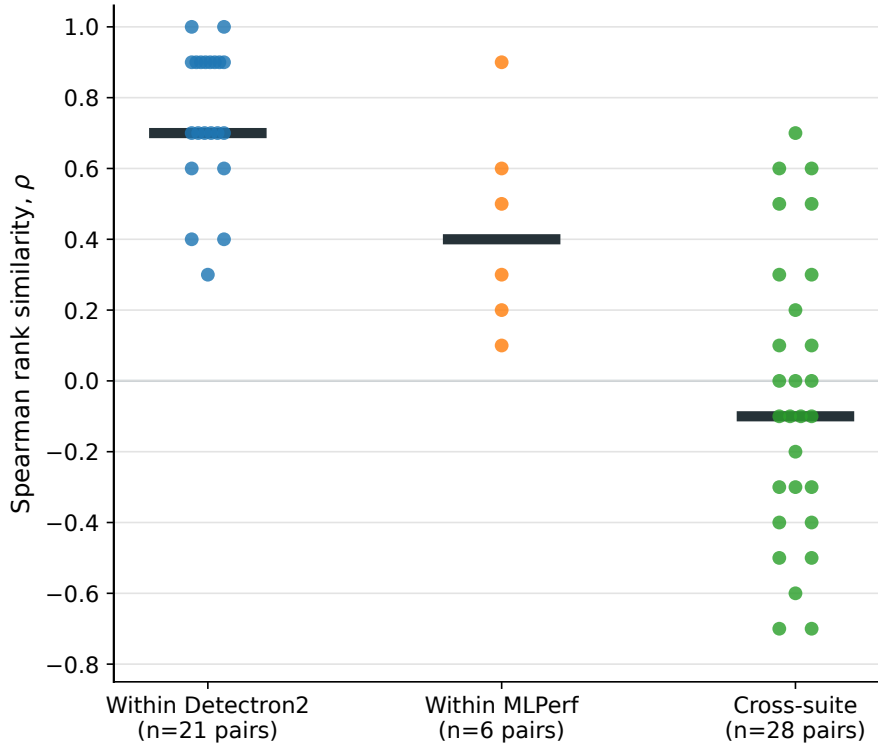


Figure 2. Spearman similarity of platform rankings within and across suites. Thick horizontal marks denote medians; pair counts are workload pairs. Points are horizontally offset within each category to expose overlapping observations with identical ρ values.

4.4. Run-to-run stability

Across the 55 cross-platform cells (35 Detectron2 and 20 MLPerf), CV ranges from 0.485% to 16.033% (Figure 4). Detectron2 occupies a narrower band, while the largest cross-platform values occur for GH200 MLPerf RetinaNet (16.033%) and BERT (14.874%). The amount of run-to-run variation depends on both system and workload.

The controlled GH200 BERT 288-CPU point is more variable still. CV is 20.233% in the primary campaign ($n = 3$), 14.166% in the independent replication ($n = 3$), and 9.860% in the diagnostic campaign ($n = 6$), and the three medians also differ. These samples were collected for different purposes, so Figure 4 presents them side by side rather than merging them into one nominal sample.

5. Discussion

5.1. Workload-dependent platform choice

The cross-platform comparison does not produce a stable global ordering. First place is split among AMD EPYC, Grace, and GH200 CPU, and rank agreement is weakest when workloads come from different suites. Prior work already shows strong model and framework sensitivity on CPUs [Liu et al. 2019, Ahn et al. 2023, Prieto et al. 2023, Pati et al. 2022]; our results show that the *platform ordering* inherits that sensitivity. The near-zero cross-suite median is best read as limited agreement between benchmark families, not as evidence that one suite is intrinsically more representative than the other.

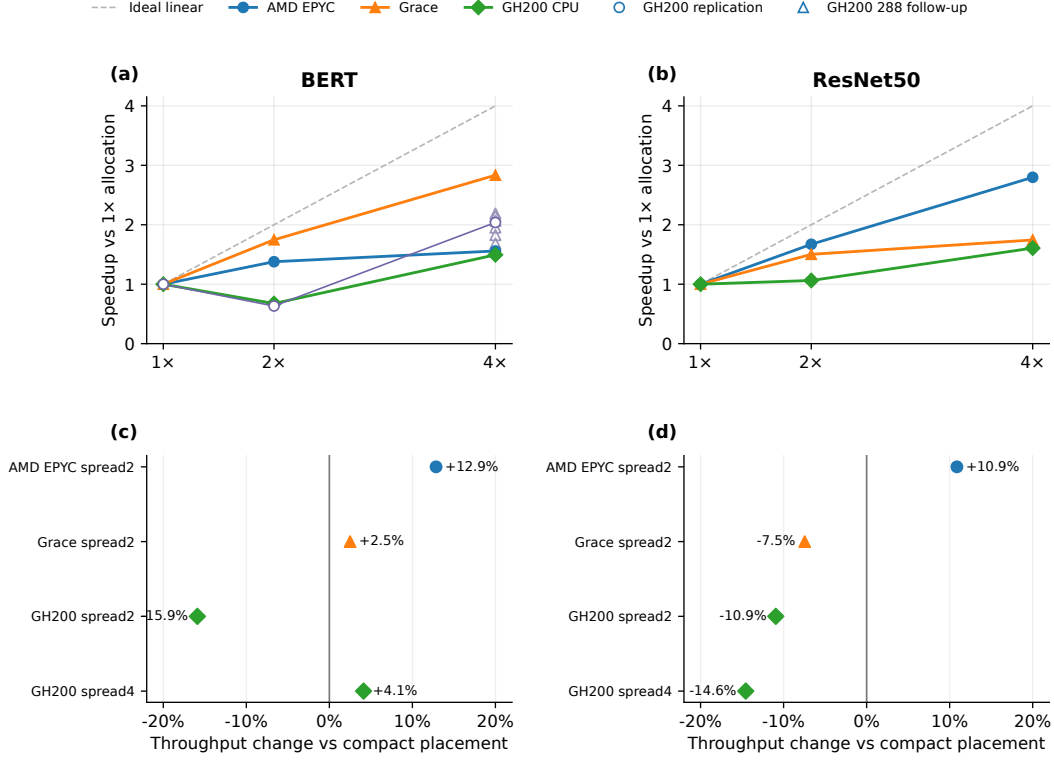


Figure 3. Controlled scaling and fixed-resource spread-versus-compact effects for BERT and ResNet50. Filled markers denote the primary campaign; open markers show additional GH200 campaigns where available.

Detectors2 and MLPerf exercise different workloads, software paths, and measurement contracts. A ranking obtained in one family consequently provides only limited evidence for another unless the ordering is shown to persist. Within Detectors2, by contrast, the winner labels are robust: changing the timed path does not change the identity of the winning system in any of the 14 predefined checks.

The MI300A result illustrates the same need for metric context. It ranks first on T/N across all workloads while using the smallest CPU allocation and without leading absolute throughput overall. That result is useful as a measure of allocation productivity, but it should not be turned into a claim about intrinsic core or architectural efficiency.

5.2. Scaling and placement are configuration-dependent

More parallel resources do not guarantee proportional speedup [Nori et al. 2021, Mebratu et al. 2021, Camargo et al. 2021]. Every tested $2\times$ and $4\times$ point is sublinear. The GH200 BERT case is stronger than diminishing returns: throughput falls by 32.5% from 72 to 144 compact CPUs in the primary campaign and by 36.7% in the independent replication. The repeated direction makes a one-off run less plausible, while the different effect size and high 288-CPU variability argue against a stronger reproducibility claim.

Placement is equally conditional. Affinity and NUMA-aware placement are known to affect parallel applications [Mazouz et al. 2011, Denoyelle et al. 2019, Dokulil and Benkner 2020], and CPU-inference studies likewise treat scheduling as an important tuning dimension [Jiang et al. 2023, Khan and Islam 2026]. In our fixed-resource

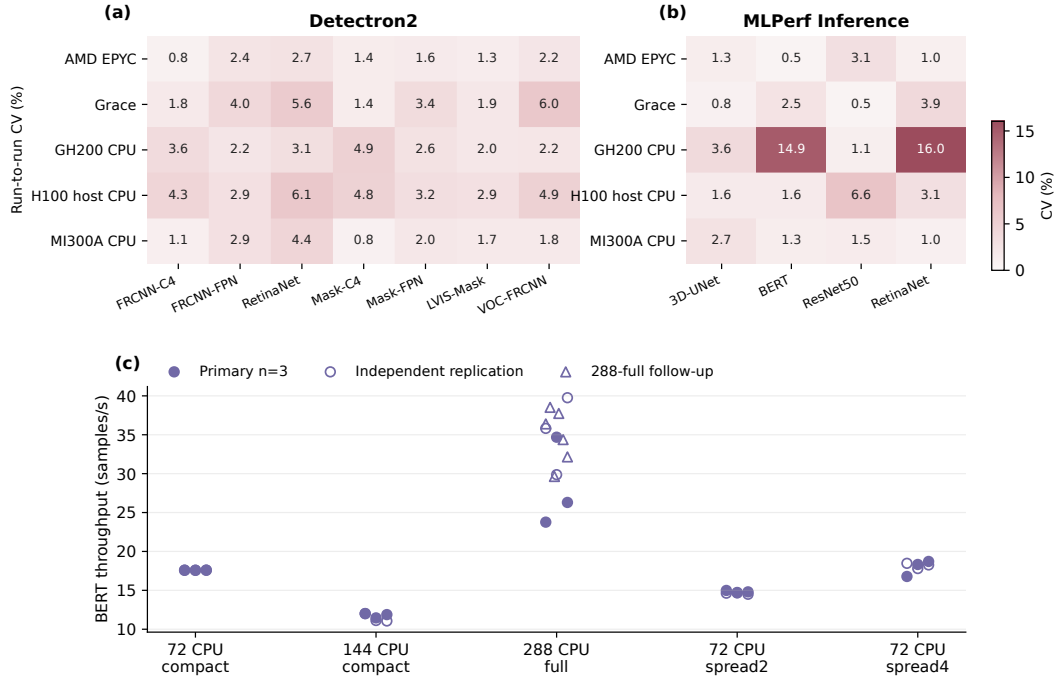


Figure 4. Cross-platform run-to-run CV for Detectron2 and MLPerf (top) and GH200 BERT observations from primary, replication, and diagnostic campaigns (bottom).

experiments, placement alone changes throughput by as much as 15.9%. The GH200 reversal is particularly illustrative: spread4 is the best 72-CPU placement for BERT in the primary campaign, but the worst for ResNet50.

A plausible class of explanations is the trade-off between locality and the memory resources reached as execution spans more CPU-bearing NUMA domains, consistent with prior placement studies [Denoyelle et al. 2019, Dokulil and Benkner 2020]. We cannot distinguish that mechanism from other runtime, cache, bandwidth, or frequency effects: CPU binding was controlled, memory placement was not explicitly bound, and the campaign did not collect the hardware-counter profiling needed for causal attribution. The data establish configuration sensitivity; identifying its microarchitectural cause remains future work.

5.3. Repeated valid runs add stability information

A valid MLPerf run passes the duration and query-count requirements, but this alone says little about how much another independently valid execution may differ [MLCommons 2025]. The repeated runs provide that second piece of information. Most cells are comparatively stable, yet the upper end of the observed CV range reaches 16.0% cross-platform and exceeds 20% in the primary GH200 BERT 288-CPU controlled point.

The follow-up campaigns also illustrate why experimental purpose matters. Systems methodology recommends preserving enough structure to distinguish planned measurements from later investigation [Hoeftler and Belli 2015, Kalibera and Jones 2013]. Merging the GH200 primary, replication, and diagnostic samples would increase the nomi-

nal n while mixing observations collected to answer different questions. Accordingly, we keep them separate and interpret the $n = 3$ controlled-cell CVs descriptively rather than as precision estimates or confidence intervals.

5.4. Scope and limitations

This is a study of five deployed HPC CPU configurations, not a controlled architecture comparison. The systems expose unequal allocations (48–288 CPUs), accelerator execution is excluded, and Detectron2 and MLPerf use different stacks, timed regions, and metrics. Uniformity applies within each benchmark family. Hardware ISA, runtime/library kernels, and memory-system configuration are part of the measured platform and are not factored into independent effects. Resource scaling can also change both CPU count and the topology spanned by the allocation; the fixed-resource compact-versus-spread experiments provide the cleaner placement comparison. The controlled campaign covers three platforms and two workloads with three primary repetitions per configuration. We do not evaluate power, energy, cost, or formal performance portability.

Accuracy serves only as a correctness guard. The current GH200 Detectron2 accuracy campaign contains one execution per selected workload, so accuracy variability is outside the scope of the study. These limitations constrain causal and architectural interpretation, not the observed configuration-level results. As a practical reporting guideline, such comparisons should report workload, absolute throughput, CPU allocation, placement, run-to-run dispersion, and measurement context, labeling normalized throughput as allocation productivity rather than architectural efficiency. Formal validation of a broader protocol remains future work.

6. Conclusion

We evaluated deep-learning inference on five deployed HPC CPU configurations with Detectron2 and MLPerf Inference, complemented by controlled scaling, placement, and repeated-run experiments. No platform leads all workloads, platform rankings agree less across suites than within them, additional CPUs do not guarantee proportional throughput, and fixed-resource placement can change its effect across workloads. Repeated valid MLPerf runs also show that validity and run-to-run stability are distinct. Together, these findings support workload- and configuration-aware reporting rather than a single platform score. Future work should extend the controlled campaign with explicit memory placement, hardware-counter profiling, and energy measurements, and test the reporting dimensions in broader studies. For heterogeneous HPC CPUs, performance is better understood as a property of the workload and execution context than as a fixed platform ranking.

Acknowledgments

The authors thank Petróleo Brasileiro S.A. (PETROBRAS) for funding this work and the Laboratório Nacional de Computação Científica (LNCC) for providing access to the HPC infrastructure used in the experiments.

Generative AI disclosure. Feynman, ChatGPT (OpenAI), Claude (Anthropic), and a custom LLM-assisted workflow were used to support literature discovery, evidence organization, drafting, language editing, and manuscript review. All experimental results, references, technical claims, and the final manuscript were checked by the authors, who take full responsibility for its content.

References

- [Ahn et al. 2023] Ahn, H., Chen, T., Alnaasan, N., Shafi, A., Abduljabbar, M., Subramoni, H., and Panda, D. K. (2023). Performance characterization of using quantization for DNN inference on edge devices. In *2023 IEEE 7th International Conference on Fog and Edge Computing (ICFEC)*, pages 1–6.
- [Alvarado Rodriguez et al. 2020] Alvarado Rodriguez, O., Dave, D., Liu, W., and Su, B. (2020). A study of machine learning inference benchmarks. In *Proceedings of the 4th International Conference on Advances in Image Processing (ICAIP 2020)*, pages 167–171.
- [Ben-Nun et al. 2019] Ben-Nun, T., Besta, M., Huber, S., Ziogas, A. N., Peter, D., and Hoeffler, T. (2019). A modular benchmarking infrastructure for high-performance and reproducible deep learning. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 66–77.
- [Camargo et al. 2021] Camargo, M. W., Serpa, M. S., Carastan-Santos, D., Carissimi, A., and Navaux, P. O. A. (2021). Accelerating machine learning algorithms with TensorFlow using thread mapping policies. In *High Performance Computing: 7th Latin American Conference, CARLA 2020, Revised Selected Papers*, volume 1327 of *Communications in Computer and Information Science*, pages 62–70.
- [Castelló et al. 2022] Castelló, A., Barrachina, S., Dolz, M. F., Quintana-Ortí, E. S., San Juan, P., and Tomás, A. E. (2022). High performance and energy efficient inference for deep learning on multicore ARM processors using general optimization techniques and BLIS. *Journal of Systems Architecture*, 125:102459.
- [Chunduri et al. 2017] Chunduri, S., Harms, K., Parker, S., Morozov, V., Oshin, S., Cherukuri, N., and Kumaran, K. (2017). Run-to-run variability on Xeon Phi based Cray XC systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2017)*, pages 1–13.
- [Cui et al. 2023] Cui, M., Papadopoulou, N., and Pericàs, M. (2023). Analysis and characterization of performance variability for OpenMP runtime. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, pages 1614–1622.
- [Denoyelle et al. 2019] Denoyelle, N., Goglin, B., Jeannot, E., and Ropars, T. (2019). Data and thread placement in NUMA architectures: A statistical learning approach. In *Proceedings of the 48th International Conference on Parallel Processing (ICPP 2019)*, pages 39:1–39:10.
- [Dokulil and Benkner 2020] Dokulil, J. and Benkner, S. (2020). NUMA-aware CPU core allocation in cooperating dynamic applications. In *2020 IEEE 34th International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 950–957.
- [Elordi et al. 2021] Elordi, U., Unzueta, L., Goenetxea, J., Sánchez-Carballido, S., Arganda-Carreras, I., and Otaegui, O. (2021). Benchmarking deep neural network inference performance on serverless environments with MLPerf. *IEEE Software*, 38(1):81–87.
- [Hodak et al. 2022] Hodak, M., Ellison, D., and Dholakia, A. (2022). Everyone is a winner: Interpreting MLPerf inference benchmark results. In *Performance Evaluation and*

Benchmarking: 13th TPC Technology Conference (TPCTC 2021), Revised Selected Papers, volume 13169 of *Lecture Notes in Computer Science*, pages 50–61.

- [Hoeffler and Belli 2015] Hoeffler, T. and Belli, R. (2015). Scientific benchmarking of parallel computing systems: Twelve ways to tell the masses when reporting performance results. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2015)*, pages 73:1–73:12.
- [Jiang et al. 2023] Jiang, J., Du, J., Huang, D., Chen, Z., Lu, Y., and Liao, X. (2023). Full-stack optimizing transformer inference on ARM many-core CPU. *IEEE Transactions on Parallel and Distributed Systems*, 34(7):2221–2235.
- [Kalibera and Jones 2013] Kalibera, T. and Jones, R. E. (2013). Rigorous benchmarking in reasonable time. In *Proceedings of the 2013 ACM SIGPLAN International Symposium on Memory Management (ISMM 2013)*, pages 63–74.
- [Khan and Islam 2026] Khan, M. A. and Islam, S. (2026). Optimizing deep learning inference on multi-core laptop CPUs with scheduling and thread affinity strategies. In *Proceedings of the 2026 ACM Southeast Conference (ACMSE 2026)*, pages 276–280.
- [Liu et al. 2019] Liu, Y., Wang, Y., Yu, R., Li, M., Sharma, V., and Wang, Y. (2019). Optimizing CNN model inference on CPUs. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 1025–1040. USENIX Association.
- [Liu et al. 2022] Liu, Z., Leng, J., Zhang, Z., Chen, Q., Li, C., and Guo, M. (2022). VELTAIR: Towards high-performance multi-tenant deep learning services via adaptive compilation and scheduling. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2022)*, pages 388–401.
- [Mazouz et al. 2011] Mazouz, A., Touati, S.-A.-A., and Barthou, D. (2011). Performance evaluation and analysis of thread pinning strategies on multi-core platforms: Case study of SPEC OMP applications on intel architectures. In *2011 International Conference on High Performance Computing and Simulation (HPCS)*, pages 273–279.
- [Mebratu et al. 2021] Mebratu, D., Hasabnis, N., Mercati, P., Sharma, G., and Najnin, S. (2021). Automatic tuning of TensorFlow’s CPU backend using gradient-free optimization algorithms. In *High Performance Computing: ISC High Performance 2021 International Workshops, Revised Selected Papers*, volume 12761 of *Lecture Notes in Computer Science*, pages 249–266.
- [Mittal et al. 2022] Mittal, S., Rajput, P., and Subramoney, S. (2022). A survey of deep learning on CPUs: Opportunities and co-optimizations. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5095–5115.
- [MLCommons 2025] MLCommons (2025). MLPerf inference v5.1 reference implementation and configuration files. Official MLCommons inference repository, tag v5.1. commit 75f297d44c5e68d24a2f7643fe1eb11e49313238.
- [Nori et al. 2021] Nori, A. V., Bera, R., Balachandran, S., Rakshit, J., Omer, O. J., Abuhatzera, A., Kuttanna, B., and Subramoney, S. (2021). REDUCT: Keep it close, keep it cool! efficient scaling of DNN inference on multi-core CPUs with near-cache compute. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 167–180.

- [Pati et al. 2022] Pati, S., Aga, S., Jayasena, N., and Sinclair, M. D. (2022). Demystifying BERT: System design implications. In *2022 IEEE International Symposium on Workload Characterization (IISWC)*, pages 296–309.
- [Prieto et al. 2023] Prieto, P., Abad, P., Gregorio, J. A., and Puente, V. (2023). Performance characterization of popular DNN models on out-of-order CPUs. In *2023 32nd International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 199–210.
- [Reddi et al. 2021] Reddi, V. J., Cheng, C., Kanter, D., Mattson, P., Schmuelling, G., and Wu, C.-J. (2021). The vision behind MLPerf: Understanding AI inference performance. *IEEE Micro*, 41(3):10–18.
- [Reddi et al. 2020] Reddi, V. J., Cheng, C., Kanter, D., Mattson, P., Schmuelling, G., Wu, C.-J., Anderson, B., Breughe, M., et al. (2020). MLPerf inference benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 446–459.
- [Vellaisamy et al. 2025] Vellaisamy, P., Labonte, T., Chakraborty, S., Turner, M., Sury, S., and Shen, J. P. (2025). Characterizing and optimizing LLM inference workloads on CPU–GPU coupled architectures. In *2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 49–61.