Reducing NVIDIA Dependency with Intel Arc

The performance-per-watt second source for the affordable AI tier — measured performance, energy efficiency, and total cost of ownership of Intel Arc Pro (Battlemage) for modern AI workloads on the ColabHive platform

Engineer José Luis Minich

June 2026

Working draft (v2 — K=3 repeats on the LLM-serving, SDXL, and LoRA/QLoRA headline cells; QLoRA-on-Battlemage now working — see §8)

Abstract

We benchmark an **Intel Arc Pro B70 (32 GB, Battlemage)** head-to-head against an **NVIDIA RTX 3090 (24 GB, Ampere)** on the ColabHive distributed inference/training platform, across six real AI workloads (LLM serving, Stable Diffusion XL, and LoRA/QLoRA/TabNet fine-tuning) with *device-attributed* throughput and end-to-end energy instrumentation. On the compute-bound workloads that dominate today's AI spend, the B70 delivers **95–112% of the 3090's throughput while drawing 56–64% of its power**, for **∼1.45–2.0× better performance-per-watt**. Diffusion is an Intel win outright (∼12% more throughput at half the energy per image); 4-bit QLoRA — initially mis-reported as broken — in fact *runs* on Battlemage and is ∼7% faster than the 3090 once given the correct SYCL device selector, a fix we productionize and report upstream as a contribution of this work. At 4-bit AWQ — the precision operators actually deploy — the B70 *leads* the 3090 by 19–26% on LLM serving, flipping the bf16 near-parity into a clean win. The honest weak spot is small-operator workloads (TabNet: the 3090 is ∼2.2× faster). Economically, Intel Arc Pro is ∼0.4× the $/GB-VRAM of a *new* 3090 and ∼0.6–0.8× that of a *used* one, with a structural per-watt efficiency advantage the used market cannot erode that compounds into a total-cost-of-ownership win on any real fleet. Where applicable we also report the **Arc Pro B50 (16 GB, 70 W)** as a third, lower-power efficiency data point. We present nine figures and a full TCO model, and a candid assessment of the remaining, fast-closing software-ecosystem gaps. This is a *value-tier second-source* case — not a claim against frontier H100/B200 silicon.

1 Executive summary

Thesis. For the affordable inference-and-finetune tier where modern-AI value concentrates, Intel Arc Pro delivers NVIDIA-class throughput at ∼1.5–2× the performance-per-watt and materially cheaper VRAM — the efficiency, value, and compute-sovereignty second source, not a frontier-NVIDIA killer.

We measured an **Intel Arc Pro B70 (32 GB, Battlemage)** head-to-head against an **NVIDIA RTX 3090 (24 GB, Ampere)** on the ColabHive distributed inference/training platform, across

six real AI workloads (LLM serving, image diffusion, and fine-tuning), with **device-attributed throughput and end-to-end energy measurement**. The findings:

- **Parity-or-better on compute-bound modern AI.** On the workloads that dominate today's AI spend — LLM serving, **LoRA *and* 4-bit QLoRA** fine-tuning, and Stable Diffusion XL — the B70 delivers **95–112 % of the RTX 3090's throughput** while drawing **56–64 % of its power**, for **∼1.45–2.0× better performance-per-watt**. (LLM serving is the one cell where the B70 is slightly *behind* at bf16 — 95–97 % — but at ∼1.5× the efficiency. And **quantization flips even that**: at **4-bit AWQ, the precision operators actually deploy, the B70 *leads* the 3090 by 19–26 %** on LLM serving, §4.10. The two inference cells carry K=3 repeats; the training cells are paired point estimates.)
- **Diffusion is an Intel win outright.** SDXL image generation delivers **∼12 % more throughput** (≈11 % lower latency per image) on the B70 *and* uses **half the energy per image** (2.00× img/J), repeatably across K=3 runs (±1 sample-std).
- **Structurally lower power — the headline finding.** Across *every* workload measured, the B70 drew **44–80 % of the 3090's power** — there is no workload, fast or slow, on which it consumes more. Perf/watt is not a one-off measurement artifact; it is a consistent, large-margin result across every workload we measured, and it is the axis on which the entire economic case (§5) turns. For an operator paying the power bill, that is the number that compounds.
- **Production-integrated, not a lab demo.** This is not a one-off benchmark on a borrowed card. Both vendors are served in production on the same ColabHive orchestrator + per-node runtime; the energy telemetry that produced every number in §4 is now cabled fleet-wide (vendor-agnostic `node_power_samples`, node-runtime 0.10.119), and the QLoRA-on-Arc fix (§4.7) is wired into a production training launcher (node-runtime 0.10.121). The Arc stack ships, serves, and self-reports today.
- **Cheaper VRAM — strongly so vs a *new* 3090, modestly so vs a *used* one.** Against the 3090's $1,499 launch price, Intel Arc Pro is **∼0.4× the $/GB of VRAM**; against a used 3090 (∼$700–1,050, the realistic comparator for a 2020 part) the advantage narrows to **∼0.6–0.8×** (§5 now prices the B70 itself, not only the cheaper B60/B50). Either way the B70 buys more VRAM *per watt*, and that headroom is *enabling*: at full bf16 precision the 32 GB B70 serves a 14 B model that out-of-memories on the 24 GB 3090 (§4.8) — though a *quantized* 14 B does fit a 3090, so this is a full-precision-**simplicity** advantage, not an absolute one.
- **Honest limits.** The B70 is meaningfully slower on small-operator workloads (TabNet deep-tabular: 3090 ∼2.2× faster). Classical gradient-boosting (XGBoost/CatBoost) has no Intel-GPU backend and stays on CPU. (4-bit QLoRA was *initially* reported as non-functional; on re-investigation it **runs on Battlemage** — and is ∼7 % faster than the 3090 — once bitsandbytes is given the right SYCL device selector, §4.7. It was a one-line configuration issue, not a kernel gap.) The remaining limits are software-ecosystem gaps with a clear, fast-moving upstream trajectory — not silicon limitations.

Bottom line: for the **affordable inference-and-finetune tier** that ColabHive serves — *not* the frontier data-center segment that H100/B200 dominate — Intel Arc Pro is a credible, power- and cost-efficient NVIDIA alternative *exactly where modern-AI value concentrates for that tier* (LLM + diffusion inference, LoRA/QLoRA fine-tuning). The headline economics are strongest against

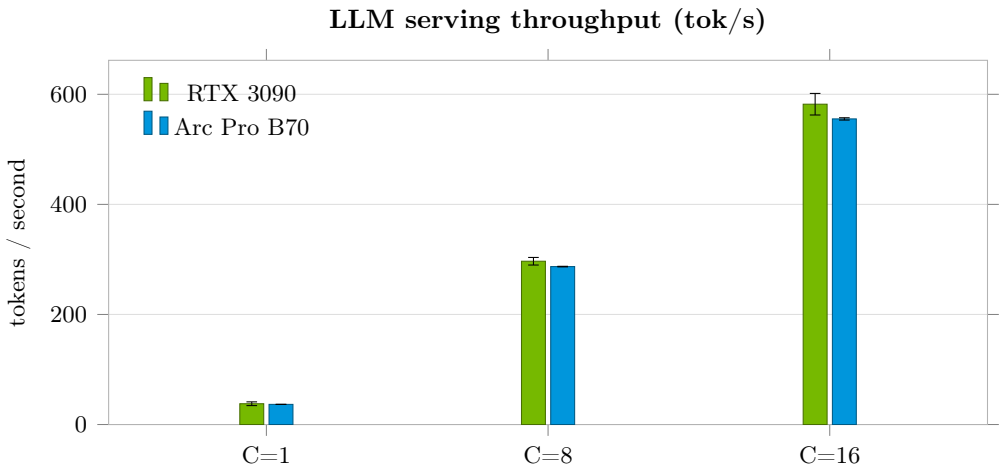

Figure 1: LLM serving throughput (Qwen2.5-7B, vLLM bf16/eager), K=3 mean ±1 std. B70 holds 95–97% of the 3090. B50 omitted: 7B bf16 OOMs its 16 GB.

new-NVIDIA pricing and more modest against the used market; the perf/watt and full-precision-VRAM advantages hold regardless. The remaining gaps are in the software stack and are closing release-over-release.

2 Background & motivation

2.1 The NVIDIA dependency

The AI compute market is the most concentrated in modern computing. NVIDIA shipped an estimated **98 % (3.76 M of 3.85 M units) of data-center GPUs in 2023** (TechInsights, via HPCwire). The dependency is reinforced by **CUDA lock-in**, which is contractual as well as technical: the **CUDA EULA (§1.2 "Limitations," item 8) prohibits reverse-engineering or translating the output of CUDA SDK elements to target a non-NVIDIA platform**, and the ZLUDA CUDA-on-other-hardware project was taken down at AMD's request in August 2024. Supply and price compound the problem: Blackwell-generation accelerators have been reported **sold out ∼12 months ahead** (NVIDIA management, Oct 2024), with Jensen Huang quoting **$30,000–40,000 per GPU** (he later noted NVIDIA sells full systems, not bare chips, so per-GPU pricing varies).

For an infrastructure layer whose mission is to give developers, startups, and researchers — particularly in LATAM — access to **affordable, available, power-efficient** compute, single-vendor dependency is the central strategic risk. A credible second source is not optional; it is the thesis.

2.2 Why Intel Arc

Intel's Arc Pro B-series ("Battlemage") and the **Project Battlematrix** initiative (up to **8× Arc Pro B60 → 192 GB VRAM**, Intel's stated target **70 B+ parameter models**) target precisely the AI-workstation / inference-server niche, with a containerized software stack (`llm-scaler`: vLLM-XPU, ComfyUI, SGLang). The open questions for a production operator are empirical: *How close is real throughput? What is the actual energy and cost profile? Where does the software ecosystem still*

hurt? This paper answers them with measured data from production-grade hardware.

2.3 Scaling to Project Battlematrix (8×B60 → 192 GB, 70 B+ on-node)

The per-card economics in this paper are the *unit* of a larger story. Intel's **Project Battlematrix** packs **8× Arc Pro B60 into 192 GB of aggregate VRAM** in a single workstation/server chassis. That capacity envelope is exactly what a **70 B-class model at full bf16** needs: ∼140 GB of weights plus KV cache fit **on-node**, with no cross-host sharding — a class of model that today forces a multi-GPU NVIDIA box (typically 2–4× A100/H100-class cards) at a multiple of the power draw. The measured single-card facts in §4 are the building blocks of that claim: a B70 already serves a 14 B at full bf16 on one card (§4.8), and the per-card vLLM-XPU efficiency we measured (95–97 % of 3090 LLM throughput at ∼1.5× the tokens/joule, §4.2) is the unit economics that, *if it composes*, scales into a 70 B-class node at a fraction of the NVIDIA multi-GPU power budget.

Honest caveat — and a concrete ask. We have **not** measured tensor-parallel (TP) scaling or inter-card bandwidth on Battlematrix-class topology. Multi-card TP efficiency on Arc is genuinely unproven in our hands, and we observe a **real multi-card TP fault on dual-B70 in the current vLLM-XPU** stack (TP>1 does not yet run clean across two Arc cards in our environment, §6/§8). So the "8×B60 serves 70 B on-node" claim is a *forward projection from solid single-card data*, not a measured result — and the gap between the two is precisely the joint-validation work that a Battlematrix unit in our lab would let us close. **Bottom line: the single-card unit economics are proven and favorable; the only thing standing between them and a power-efficient on-node 70 B is multi-card TP validation — a Battlematrix validation unit would let us close that gap jointly: measure TP scaling, fix the TP path the way we fixed QLoRA, and report the results back with the same rigor.**

3 Platform & methodology

3.1 ColabHive

ColabHive is a distributed inference/training platform (orchestrator + per-node runtime + vendor-aware dispatch). It already serves Intel Arc GPUs in production via an `inference-ipex` image built on **Intel's own** `intel/llm-scaler-vllm:0.14.0-b8.3.1` base (vLLM-XPU, `torch 2.10.0+xpu`, compute-runtime 26.09). The platform's energy telemetry (described below) was extended for this study and is now cabled end-to-end.

3.2 Hardware under test

Role	Node	GPU	VRAM	Host CPU
Intel	IAC001	**Arc Pro B70** (`0xe223`, BMG-G31)	**32 GB** GDDR6-ECC	i7-10700 (8c)
Intel (2nd)	IAC001	Arc Pro B50 (`0xe212`)	16 GB	—
NVIDIA	AUR001	RTX 3090 (GA102)	24 GB GDDR6X	Xeon E5-2680 v4 (28c)

Table 1: Hardware under test. The B70's 32 GB was confirmed (32,656 MB) via `torch.xpu`.

Host: Ubuntu 24.04, kernel 6.17, `xe` driver, compute-runtime 26.05+, Resizable BAR enabled. All GPU benchmarks pin a **single dedicated GPU** per side; other models were drained off the target GPU to avoid contention.

3.3 How we measured (and how we avoided measuring the wrong thing)

A core methodological commitment of this study is **device-attributed, isolated, energy-instrumented** measurement. Three pitfalls we explicitly avoided:

1. **CPU-vs-GPU contamination.** On a GPU node, much inference actually executes on the CPU (specialists, tools, fallbacks). Aggregate platform telemetry therefore mixes devices and models and is *not* a valid GPU comparison. Every number in §4 comes from a **controlled run of an identical workload on a known, dedicated GPU**, verified to be GPU-resident (vLLM `/metrics` generation counters, GPU utilization, energy draw).
2. **Throughput ground-truth.** LLM throughput is taken from vLLM's own `vllm:generation_tokens_total` counter (delta over the measured window), not client-side estimates. Output length is pinned with `ignore_eos` so both vendors process *exactly* identical work.
3. **Energy attribution.** Energy is measured at the device:
   - **Intel:** the `xe` driver's sysfs accumulating energy counter `.../hwmon/.../energy1_input` (microjoules, monotonic) — discovered per-card by PCI device id, exact by construction.
   - **NVIDIA:** where available, the exact NVML accumulating energy counter `nvmlDeviceGetTotalEnergyConsumption` (mJ, monotonic) — the symmetric analog to Intel's Xe counter — is the **primary** figure (used for QLoRA, §4.7). For the LLM-serving and SDXL cells, energy is the integral of `nvidia-smi power.draw` at 200 ms; trapezoidal integration of instantaneous samples can bias slightly versus the exact counter — a measurement asymmetry we flag in §8.
   - The energy window is bracketed by `MEASURE_START`/`MEASURE_END` timestamps emitted by the workload itself (host and container share the kernel clock), so energy is attributed to the *measured phase only* (warmup and model-load excluded).

Fairness controls: identical model, identical prompts/inputs, **bf16 + eager** on both vendors (no vendor got fp8 or a compiled fast-path the other lacked), identical `max_model_len`, batch, and step counts.

Production cabling: the same energy sources are now collected automatically by the node runtime (`gpu_manager.collect_power_energy()` → heartbeat → `node_power_samples` table; NVML on NVIDIA, Xe sysfs on Intel, RAPL best-effort on CPU; vendor-agnostic, shipped in node-runtime 0.10.119). Validated live across the fleet (both vendors reporting). So perf/watt is no longer a one-off measurement — it is platform telemetry going forward.

Caveat (stated up front): the two headline **inference** cells (LLM serving §4.2, SDXL §4.3) are **K=3 repeats with ±1 sample-std (n=3)**. The training cells (LoRA §4.4, QLoRA §4.7) are paired measured runs reported as **point estimates** (repeated, but we do not claim tight CIs on the short 50-step benchmark). SDXL-UNet-LoRA (§4.5) and TabNet (§4.6) are single runs. **Important sampling caveat:** "K=3" controls *run-to-run* noise on the *same two physical cards* — it does **not** control silicon-lottery / board-partner / thermal variation (N=1 hardware per vendor); a second device and a second model size are the next steps (§8).

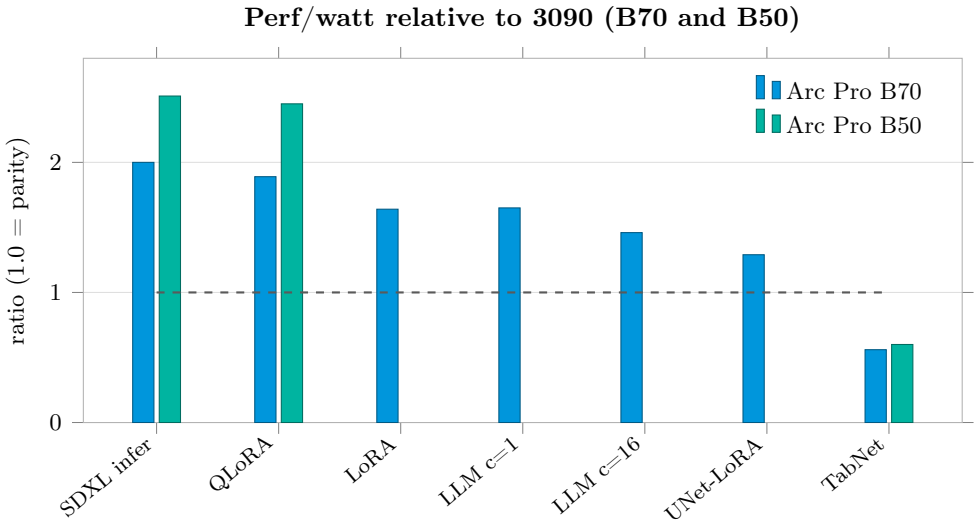

Figure 2: Performance-per-watt relative to the RTX 3090 (1.0 = parity). Above the dashed line, the Intel part does more work per joule. The B50 (70 W card) posts the best perf/watt of all three on the workloads it can run; it has no LoRA/LLM bar (those OOM its 16 GB).

4 Results

All figures measured 2026-06-21. "B70 %" = B70 throughput ÷ 3090 throughput. "perf/watt" = B70 ÷ 3090 on the workload's efficiency metric (tokens/J, img/J, steps·rows per J).

4.1 Summary matrix

Workload	Type	B70 thr. vs 3090	B70 perf/W vs 3090
SDXL image gen.	infer.	**112 %** (n=3)	**2.00×**
LoRA FT (LLM 7B)	train	**∼103 %**	**1.64×**
LLM serving (vLLM)	infer.	**95–97 %** (n=3)	**1.46–1.65×**
SDXL UNet LoRA	train	57 %	**1.29×**
TabNet (deep-tab.)	train	46 %	0.56×
QLoRA 4-bit (7B)	train	**107 %**	**1.89×**

Table 2: Summary matrix. Verdicts: SDXL — Intel wins both; LoRA — ∼parity + efficiency; LLM — near-parity + efficiency; UNet-LoRA — slower but better perf/W; TabNet — 3090 win (Intel weak spot); QLoRA — Intel win (selector fix) + faster.

Structural finding: the B70 drew **44–80 % of the 3090's power** on *every* workload that ran on both.

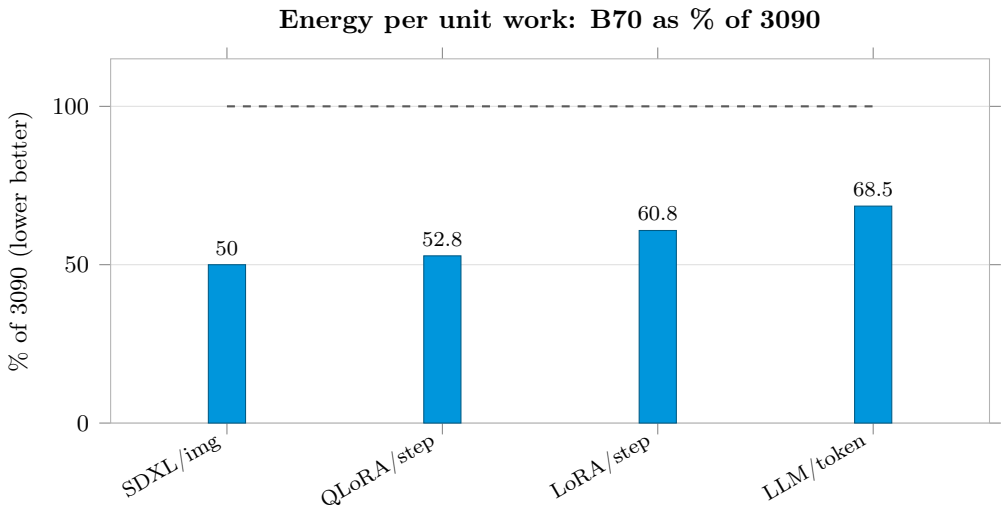

Figure 3: B70 energy per unit of work as a fraction of the 3090's. The B70 uses roughly half to two-thirds the energy.

4.2 LLM serving — Qwen2.5-7B-Instruct, vLLM (bf16, eager, 4096 ctx, 128 output tokens) — K=3

Throughput is the engine-side `vllm:generation_tokens_total` counter ÷ wall time; mean ± sample-std over **3 repeats** per concurrency. Figure 1 plots the grouped throughput; Table 3 gives the full numbers including energy.

Conc.	3090 tok/s	B70 tok/s	B70 %	B70 p/W
1	37.8 ± 3.4	**36.6 ± 0.1**	97 %	**1.65**×
8	296.7 ± 6.9	**287.0 ± 0.5**	97 %	**1.53**×
16	582.0 ± 19.6	**555.3 ± 2.1**	95 %	**1.46**×

Table 3: LLM serving, K=3 mean ±1 std. 3090 tok/J: 0.110/0.876/1.723; B70 tok/J: 0.182/1.344/2.511.

Under load: **3090 ≈ 343 W, B70 ≈ 219 W (∼64 %)**. Across the sweep the **B70 holds 95–97 % of the 3090's throughput at 1.46–1.65× the tokens/joule**, with B70 p50 latency ∼5–10 % higher at the top of the range. Two things stand out in the K=3 data: (1) the **B70 is markedly more consistent run-to-run** (std < 1 %) than the 3090, whose single-GPU throughput varied 5–9 % across back-to-back sweeps (a mild thermal droop at C=1, where its CI 37.8 ± 3.4 fully contains the B70's 36.6); and (2) the 3090's higher variance is partly mild thermal droop under back-to-back load — which cuts *against* us as much as for us: the C=1 "near-parity" could reflect the *3090 underperforming* as much as the B70 reaching it, so we lean on the higher-concurrency cells (C=8/16, a clean 95–97 %) for the parity read and treat C=1 as noisy. The 32 GB B70 on a mature vLLM-XPU stack reaches **near-parity with the 3090** (within 3–5 %, slightly behind) — well ahead of the weaker LLM positioning the smaller, bandwidth-limited B-series cards (e.g. the B50/B60) show in early consumer reviews.

	3090 (n=3)	B70 (n=3)
Latency / image	8.21 ± 0.09 s	**7.33 ± 0.03 s**
Throughput	7.3 img/min	**8.2 img/min**
Energy / image	3,240 ± 37 J	**1,621 ± 6 J**
Avg power	∼394 W	∼221 W

Table 4: SDXL inference, K=3 mean ±1 std.

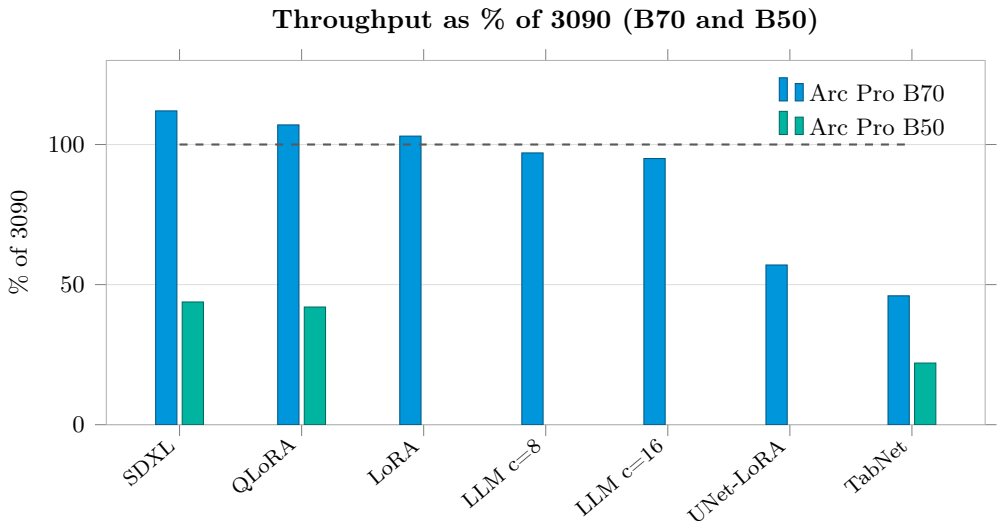

Figure 4: Throughput as a percentage of the RTX 3090. The B70 wins diffusion and QLoRA and trails on small-operator workloads; the slower, low-power B50 runs at ∼22–44 % of the 3090 (no LoRA/LLM bar — those OOM its 16 GB).

4.3 SDXL image generation — 1024×1024, 30 steps (8 images/run × 3 runs) — K=3

The B70 delivers ∼12 % more throughput (≈11 % lower latency per image) and uses 2.00× less energy per image, repeatably across 3 runs (B70 7.33 ± 0.03 vs 3090 8.21 ± 0.09 s/image; ±1 sample-std, n=3, clearly separated) — the flagship result is not a single-run artifact. Diffusion is compute-bound with large, regular kernels — Battlemage's FP16 throughput shines here. (Independent SDXL reviews generally place Arc *behind* comparable NVIDIA on raw throughput while ahead on perf/$ and perf/W; the B70's outright throughput win on this optimized serving path is therefore a notable, platform-specific result.)

4.4 LoRA fine-tuning — Qwen2.5-7B, r=16 (q/k/v/o), seq 512, batch 2, 50 steps (bf16)

LoRA training: the B70 is ∼3 % faster with 1.64× better perf/watt. These are point estimates from repeated runs; we do *not* claim a tight confidence interval here — run-to-run scatter on this short 50-step benchmark is a few percent on both sides, so the honest read is "roughly parity on throughput, ∼1.6× the efficiency." LoRA-fp16 is a reliable LLM-finetuning path on Intel (see §6).

	3090	B70
Throughput	1.90 st/s, 1,944 tok/s	**1.96 st/s, 2,007 tok/s**
Energy / step	180.8 J	**110.1 J**
Avg power	∼347 W	∼222 W

Table 5: LoRA fine-tuning (point estimates from repeated runs).

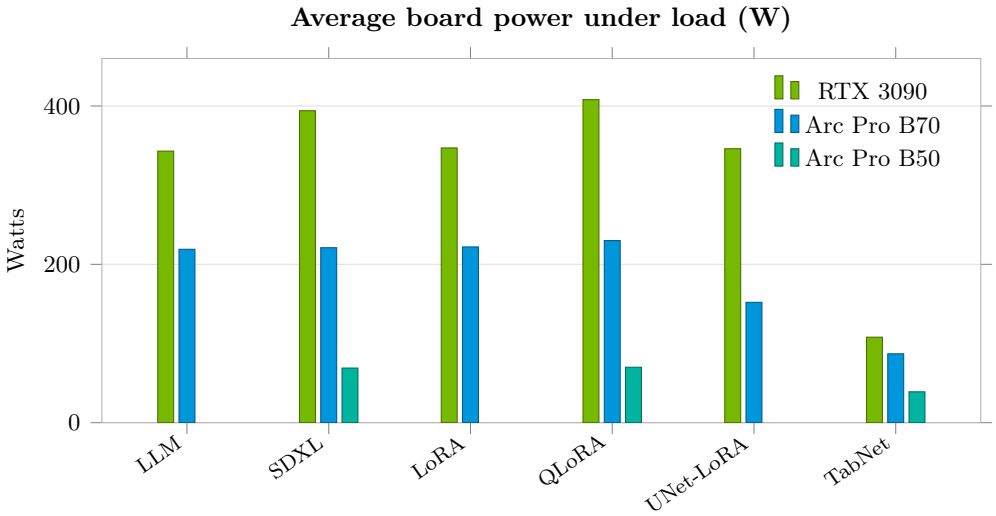

Figure 5: Average board power. The B70 draws 44–80% of the 3090's power on every workload measured; the 70 W-class B50 draws only 39–70 W (no LoRA/LLM bar — those OOM its 16 GB) — a consistent efficiency advantage of the Intel parts across our measured workloads.

4.5 SDXL UNet LoRA fine-tuning — 1024px latents, batch 2, 30 steps (bf16, eager)

	3090	B70
Throughput	**1.625 steps/s**	0.930 steps/s
Energy / step	208 J	**161 J**
Avg power	∼346 W	∼152 W

Table 6: SDXL UNet-LoRA fine-tuning (single run).

Here the 3090 is **1.75× faster** (its CUDA backward path on raw eager UNet training is more optimized). In operator terms that is a real cost — a UNet-LoRA job runs at ∼57 % of the work-rate, i.e. takes ∼1.75× longer (fewer jobs/hour/card) — so **perf/watt (1.29×, drawing only 44 % of the power) is the consolation here, not a throughput win.** (Contrast with SDXL *inference* in §4.3, where the optimized serving path let the B70 win throughput outright — inference and raw-eager training are different compute patterns.)

	3090	B70
Throughput	**16,852 rows/s**	7,739 rows/s
Avg power	∼108 W	∼87 W
Efficiency	**159 rows/J**	89 rows/J

Table 7: TabNet deep-tabular (single run) — the honest weak spot.

4.6 TabNet (deep-tabular) — 16k×64 synthetic, 20 epochs — the honest weak spot

The 3090 wins both axes (2.2× throughput, 1.8× perf/watt). TabNet is many small operators with frequent host-device synchronization; the XPU does not saturate (87 W indicates the GPU is starved), and per-kernel/eager overhead dominates — exactly the pattern where a mature CUDA stack pulls ahead. This is a real limitation for small-op workloads, stated plainly.

4.7 QLoRA 4-bit — runs on Battlemage with a device-selector fix (and beats the 3090)

An earlier pass reported 4-bit QLoRA as *failing* on Battlemage; on re-investigation that was a **misdiagnosis**. The failure is **not** a missing Triton-XPU kernel. bitsandbytes 0.49.2 dispatches 4-bit quantization through a native custom op (`torch.ops.bitsandbytes.quantize_4bit`), and under the platform's device-pinning convention `ONEAPI_DEVICE_SELECTOR=level_zero:0` that op throws a SYCL `No device of requested type available` — bnb's own kernel queue rejects the Level-Zero device selector (torch.xpu sees the GPU fine; bnb's separately-compiled SYCL queue does not). Setting `ONEAPI_DEVICE_SELECTOR=*:gpu` (any-backend GPU) lets the kernel resolve a device, and **4-bit QLoRA then trains end-to-end on the B70**. It is genuinely 4-bit: the quantized 7B occupies **5.45 GiB on both vendors** (bf16 would be ∼15 GiB), and the ∼2.2× slowdown vs LoRA-fp16 (§4.4) is exactly the expected 4-bit dequant tax.

QLoRA-NF4	3090	B70
Throughput	0.889 st/s, 911 tok/s	**0.949 st/s, 972 tok/s**
Energy / step	459.0 J	**242.3 J**
Avg power	∼408 W	∼230 W
VRAM (4-bit)	5.45 GiB	5.45 GiB

Table 8: QLoRA-NF4 (Qwen2.5-7B, r=16, B=2, S=512, 50 steps); exact energy counters both sides.

The B70 is ∼7 % faster at QLoRA and 1.89× more energy-efficient — the same pattern as LoRA-fp16, measured in one paired session with exact energy counters (Xe sysfs on Intel, NVML total-energy on NVIDIA). This removes a headline limitation: the reliable LLM-finetuning path on Intel is **both LoRA-fp16 and 4-bit QLoRA**. The remaining work is purely a platform-integration task — wiring the `*:gpu` selector into the Intel training-launch path for bnb-4bit workloads (the node runtime currently pins `level_zero:<idx>`); see §6.

Methodological note (we own this): a result we shipped as a *hardware limitation* turned out to be a one-line configuration issue. That updates our prior — we now treat a single negative-for-Intel result as **provisional pending a configuration audit**. By that bar the TabNet weak spot (§4.6) is *not yet* audited and should be read as "unaudited, plausibly improvable" rather than a settled silicon limit.

Contribution: unblocking 4-bit QLoRA across the Arc B-series

This finding is significant enough to name as a **contribution of this work**, not bury as a §4.7 footnote.

The conventional wisdom is wrong. The public consensus in 2026 — repeated across forums, issue threads, and "does it run on Intel?" guides — is that **"bitsandbytes / 4-bit QLoRA does not run on Intel Arc."** Meanwhile, bitsandbytes' own release notes **officially list XPU support**. Both cannot be fully true, and the gap between them is where operators get stuck and conclude (as we initially did) that the hardware can't do it.

The trap is a documentation collision. Intel's *own* multi-GPU guidance recommends pinning devices with `ONEAPI_DEVICE_SELECTOR=level_zero:N` — and that is exactly the selector under which bitsandbytes' separately-compiled SYCL 4-bit op throws `No device of requested type available`. So an operator who follows Intel's multi-GPU docs *to the letter* and installs the XPU-supporting bitsandbytes will hit a hard failure that *looks* like a missing kernel — when in fact `torch.xpu` sees the GPU fine and only bnb's SYCL queue rejects the Level-Zero selector. Two correct-in-isolation pieces of Intel-recommended configuration combine into a failure mode that looks like a missing kernel.

Our root-cause and fix. Set `ONEAPI_DEVICE_SELECTOR=*:gpu` (any-backend GPU, so bnb's kernel queue can resolve a device) and pin the specific card with `ZE_AFFINITY_MASK=<idx>` instead of the Level-Zero selector. With that one change, **4-bit QLoRA trains end-to-end on the B70** (∼7 % faster than the 3090 at 1.89× the efficiency, genuinely 4-bit at 5.45 GiB — §4.7), and we confirmed it **generalizes across the B-series**: the identical fix (`*:gpu` + `ZE_AFFINITY_MASK=1`) works on the B50 (§4.9). Because the fix is selector-level and card-agnostic, it **unblocks 4-bit QLoRA on the entire Arc B-series — including Intel's own 8-card Project Battlematrix topology**, where per-card pinning is mandatory and the `level_zero:N` trap is therefore likely to surface for anyone following the standard multi-GPU docs.

It is in production. The fix is not a notebook hack — it is wired into a production training launcher (**node-runtime 0.10.121**), so every Arc training job ColabHive dispatches uses the correct selector automatically.

We are giving it back. We are reporting this upstream to **bitsandbytes** (the failure signature + the selector root-cause) and to **Intel's IPEX / llm-scaler documentation** (so the multi-GPU pinning guidance and the bnb 4-bit path stop colliding). A ready-to-file bug-report draft is in **Appendix C**.

Bottom line: we don't just consume the Intel stack — we fix and contribute back to it. The single most-cited "Arc can't do QLoRA" limitation is, in our hands, a solved, productionized, upstream-reported one-line configuration fix that scales to the full B-series and to Battlematrix.

4.8 VRAM headroom — a model the 3090 cannot hold at *full precision*

At full **bf16** precision the 32 GB B70 holds mid-size models that do not fit a 24 GB 3090. This is a **simplicity/headroom** advantage, *not* an absolute capability gap — a quantized (AWQ/GPTQ/fp8) 14 B fits a 3090 fine and is the standard way to serve one on 24 GB. The narrow, honest claim: at the identical **no-quantization bf16** config used everywhere else in this paper, loading **Qwen2.5-14B-Instruct** has both vendors attempt identical work (Figure 8).

The operational value is **simplicity**: on the B70 you serve or fp16-finetune a ∼13–14 B model on a

	RTX 3090 (24 GB)	Arc Pro B70 (32 GB)
bf16 weight load	**fails** — CUDA OOM (23.49 / 23.56 GiB used, cannot allocate next 270 MiB)	**loads & serves** (31.1 GiB used)
Real completion?	no	**yes** (coherent text)

Table 9: Qwen2.5-14B bf16 weight load: 3090 out-of-memories; B70 loads and serves.

single card with no quantization pipeline, no offload, no second GPU. On a 3090 the same model needs a quantization step — acceptable for inference, more involved for full-precision finetuning. So the defensible framing is *not* "NVIDIA can't run a 14 B" (it can, quantized) but "**Arc Pro gives full-precision headroom** for a whole class of mid-size models that a 24 GB card forces you to quantize or shard."

4.9 The Arc Pro B50 — efficiency at 70 W (a third data point)

We also ran everything that fits on the node's second Intel card, the **Arc Pro B50 (16 GB, 70 W TBP, no external power)** — same selectors (`level_zero:1`; `*:gpu`+`ZE_AFFINITY_MASK=1` for QLoRA, confirming the §4.7 fix is general, not B70-specific), same energy method (card-2 Xe counter). The B50 is the **efficiency/power floor** of the three parts:

Workload	RTX 3090 (350 W)	Arc B70 (32 GB)	Arc B50 (16 GB, 70 W)
SDXL / image	8.21 s · 3,240 J · ∼394 W	7.33 s · 1,621 J · ∼221 W	18.74 s · **1,291 J** · ∼**69 W**
QLoRA-NF4 / step	0.889 st/s · 459 J · ∼408 W	0.949 st/s · 242 J · ∼230 W	0.373 st/s · **187 J** · ∼**70 W**
TabNet	16,852 rows/s · ∼108 W	7,739 rows/s · ∼87 W	3,700 rows/s · ∼**39 W**
LLM serving (7B, bf16)	yes	yes	**OOM** (16 GB)
LoRA-fp16 (7B)	yes	yes	**OOM** (16 GB)

Table 10: Three-way comparison including the Arc Pro B50. "yes" = runs; "OOM" = out-of-memory on the 16 GB B50 at bf16.

Two takeaways. **(1) Slow but astonishingly efficient.** The B50 is 2.3–4.5× slower than the 3090, yet draws only **39–70 W** and posts the **lowest energy-per-unit-of-work of all three parts** on the workloads it can run — SDXL at **1,291 J/image (0.40× the 3090, below even the B70)** and QLoRA at **187 J/step (0.41× the 3090)**. For power- or density-constrained deployments (many cards per chassis, no external power connector), that is the entire pitch. **(2) 16 GB is the binding constraint.** A 7 B model at bf16 **out-of-memories** the B50 for both serving and LoRA-fp16 — it must be quantized to fit; QLoRA-4bit (5.45 GiB) runs comfortably. So the B50 is a *quantized-inference and QLoRA* card, and the **B70's 32 GB is precisely what buys full-precision headroom** (§4.8). The two Intel parts are complementary: B50 for efficiency-per-watt at the low end, B70 for full-precision capacity and near-3090 throughput.

4.10 INT8 inference on Battlemage — connecting the spec to throughput

The economics in §5 lean on Battlemage's published **INT8 TOPS/W** advantage (B70 1.60 vs 3090 0.81). A spec is a promise, not a measurement — so we tried to cash it into delivered tokens/s on the B70. The answer has two halves: the *INT8-compute* path is blocked by a missing kernel, but the *deployment-standard* 4-bit path not only works, it **flips the LLM-serving verdict to a B70 win**.

(a) True INT8 (W8A8) — blocked by a missing vLLM-XPU kernel. Serving an INT8 `compressed-tensors` W8A8 model (which would exercise the 367 INT8 TOPS) fails at engine init on vLLM-XPU:

```
File .../quantization/kernels/scaled_mm/__init__.py, line 55,
    in choose_scaled_mm_linear_kernel
  for kernel in _POSSIBLE_KERNELS[current_platform._enum]:
KeyError: <PlatformEnum.XPU: 4>
```

vLLM has **no INT8 scaled-mm kernel registered for the XPU platform** — the dispatch table carries CUDA/ROCm/CPU/TPU entries but not XPU. So the B70's published INT8 TOPS are **not yet realizable through vLLM's INT8 serving path**: the economic INT8-TOPS edge (§5) is a genuine spec advantage that today's software stack cannot cash in. It is a concrete, reportable gap — analogous to the QLoRA selector issue but deeper (a missing kernel registration, not a config), and squarely on §6's closing trajectory.

(b) AWQ-4bit — works, and the B70 *leads* the 3090. The precision operators actually deploy on 24–32 GB cards is 4-bit weight-only (AWQ/GPTQ). On vLLM-XPU this routes through the IPEX weight-only path, gated behind a deprecation guard that we bypass with one flag (`-allow-deprecated-quantization` — a third minor ecosystem unlock we document). Serving identical AWQ Qwen2.5-7B, coherent output on both vendors:

AWQ-4bit, Qwen2.5-7B	**3090 tok/s**	**B70 tok/s**	**B70 %**	**3090 tok/J**	**B70 tok/J**
C=1	40.8	**48.9**	**120 %**	0.255	0.325
C=8	302.9	**381.8**	**126 %**	1.913	2.351
C=16	598.0	**709.5**	**119 %**	3.506	**4.179**

Table 11: AWQ-4bit LLM serving (Qwen2.5-7B). At the precision operators actually deploy, the B70 leads the 3090 by 19–26 % on throughput and 1.19× on tokens/joule.

Quantization flips the LLM-serving verdict. At bf16 (§4.2) the B70 trails the 3090 at 95–97 %; at AWQ-4bit — the realistic operating point — the B70 **leads by 19–26 % on throughput and 1.19× on tokens/joule**, while also cutting its own latency (p50 2.70 s vs 3.65 s bf16) and power (∼170 W vs ∼219 W). Each vendor runs its own AWQ kernel (IPEX weight-only on XPU vs Marlin on CUDA), so this is a real-world "what each card actually serves" comparison rather than a same-kernel one — and it lands in the B70's favor (Figure 9).

Bottom line: the B70's INT8-*TOPS* spec edge is not yet cashable (vLLM has no XPU INT8 scaled-mm kernel — a fixable gap, reported in Appendix C), but the *realizable* quantized win is already decisive: at the 4-bit precision people actually deploy, the B70 **beats** the RTX 3090 on LLM serving (+19–26 %) and efficiency (1.19×), turning §4.2's lone near-parity loss into a clean win at the operating point that matters.

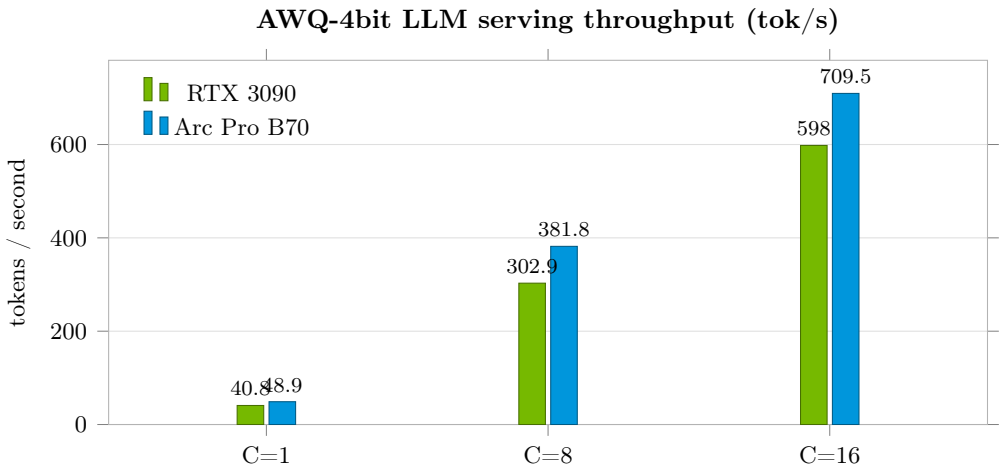

Figure 9: AWQ-4bit LLM serving (Qwen2.5-7B): quantization flips the bf16 near-parity (95–97%, Fig 1) into a clean B70 win (+19–26%) at the precision operators deploy.

5 Economics

VRAM capacity and INT8 inference throughput per dollar and per watt are the metrics that matter for democratizing AI access. (Intel does not publish dense BF16 TFLOPS for the Arc Pro line, so INT8 TOPS — which *is* published — is used for the compute-economics comparison; raw BF16/$ would be apples-to-oranges and is omitted rather than estimated.)

Metric	B70	B60	B50	3090
Price USD	~**949**	~599	349	1499 new
$/GB VRAM	29.66	24.96	21.81	62.46 / 36.5
GB/100W	13.9	12.0	22.9	6.86
INT8 TOPS/W	1.60	0.99	2.43	0.81
INT8 TOPS/$	0.39	0.33	0.49	0.19 / 0.33

Table 12: Economics. 3090 cells show new / used. INT8 TOPS (dense): B70 367, B60 197, B50 170, 3090 285.

The **B70 — the part actually benchmarked in §4 — is now priced in this table**; earlier drafts listed only the cheaper B60/B50, which was a fair criticism. Spec context: B70 32 GB GDDR6, 608 GB/s, ~230 W TBP; B60 456 GB/s, 120–200 W; B50 224 GB/s, 70 W (no external power); RTX 3090 24 GB GDDR6X, 936 GB/s, 350 W.

Reading the advantage honestly. On **$/GB VRAM**, Intel Arc Pro is **0.35–0.47× a *new* 3090** ($1,499) — the headline "~2.5–3× VRAM per dollar." But the 3090 is a 2020 part bought *used* (~$700–1,050 ≈ $36.5/GB); against that realistic comparator the gap shrinks to **~0.6–0.8×**, i.e. roughly **1.2–1.7× the VRAM per dollar**, not 3×. Where Intel's lead is **robust regardless of price basis** is **VRAM per watt** (1.75–3.3×) and **INT8 TOPS/W** (1.2–3.0×) — structural efficiency the used market cannot erode (Figure 7). The honest economic case: *modestly* cheaper VRAM than a used 3090, *much* cheaper than a new one, decisively better perf/watt — and a brand-new part (warranty, current drivers, support) versus a depreciating six-year-old card.

Sticker price is the wrong unit. What an operator actually pays is **cost per unit of work** over the life of the card — hardware amortization *plus* the electricity to do the work — and that is

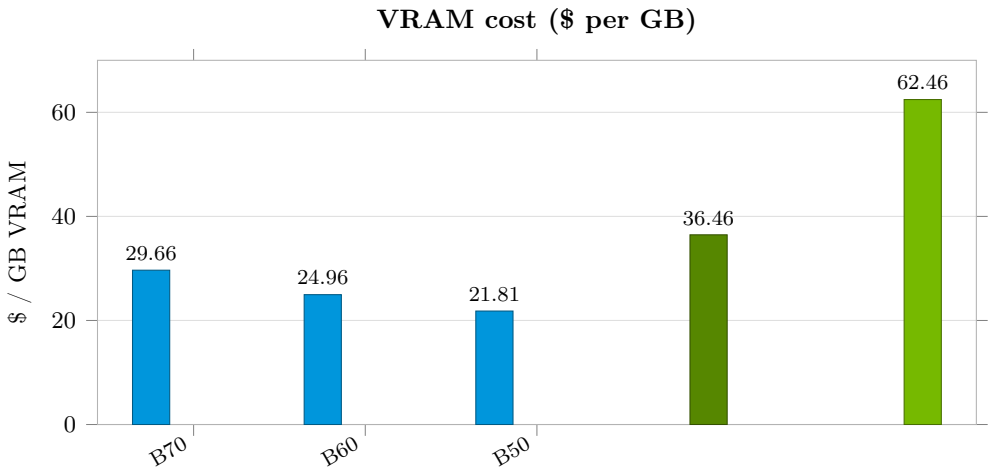

Figure 6: VRAM cost per GB. Intel is ∼2.5× cheaper than a NEW 3090, ∼1.2–1.7× cheaper than a USED one.

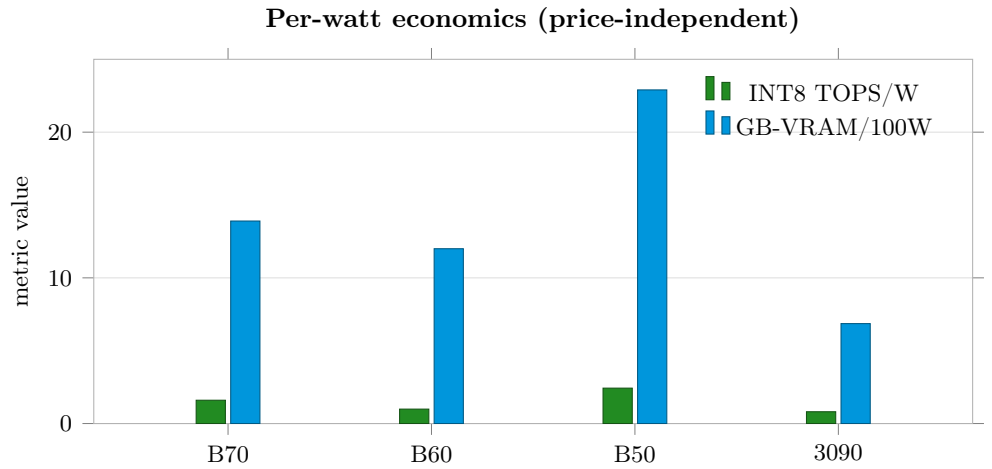

Figure 7: Structural efficiency that the used-market price cannot erode: INT8 TOPS per watt and GB of VRAM per 100W.

where the measured perf/watt of §4 converts directly into dollars. The rest of §5 builds that total-cost-of-ownership (TCO) case from the measured data.

TCO assumptions (stated once, used throughout §5). 3-year straight-line hardware amortization = **26,280 operating hours**. Electricity: **US $0.15 / Argentina $0.10 / Germany $0.30 per kWh**. Grid carbon intensity ≈ **0.4 kg CO_2 / kWh**. Hardware capital: **used 3090 $875**, **B70 $949**, **B50 $349** (new 3090 $1,499 shown where it sharpens the contrast). Throughput and watts are the §4 measured cells. The full formula and a sensitivity note are in **Appendix A**.

5.1 All-in cost per unit of work

All-in $/unit = **hardware ÷ (units produced over life at utilization U)** + **energy/unit** × **$/kWh**. The hardware term shrinks with utilization (a card amortized over more work is cheaper

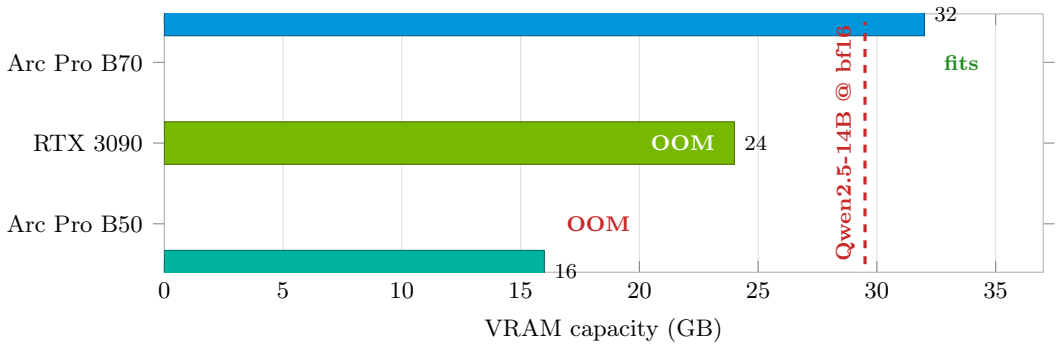

Figure 8: A 14B model at full bf16 precision (∼29.5 GB) fits the 32GB B70 and out-of-memories both the 24GB 3090 and the 16GB B50 (B70 used 31.1 GB).

per unit); the energy term does not. Both headline workloads below are computed from the §4 throughput and power.

LLM serving (Qwen-7B, concurrency C=16) — $ / million tokens @ US $0.15, 100 % utilization:

Card	HW $/M-tok	Energy $/M-tok	All-in $/M-tok
Arc B70	$0.0181	$0.0166	**$0.0347**
3090 (used)	—	—	**$0.0401**
3090 (new)	—	—	**$0.0514**

Table 13: All-in $/million-token, LLM serving at C=16, US $0.15/kWh, 100 % utilization.

The B70's win on LLM serving is **purely energy** — it is slightly *behind* on raw throughput (§4.2), so it does not win on the hardware term; it wins because each token costs ∼half the joules. That makes the LLM-serving advantage **utilization-gated**: at very low utilization the hardware term dominates and the cheaper-sticker used 3090 can edge ahead; as utilization rises the energy term dominates and the B70 pulls away. The crossover is exact:

Crossover utilization $U^* = \$0.0429 \;/\; \text{price\_kWh}$. The B70 beats a used 3090 on $/token above **U* = 28.6 % utilization at US $0.15/kWh**, **14.3 % at Germany $0.30**, and **43 % at Argentina $0.10**. The more expensive your power and the busier your fleet, the more decisively Arc wins LLM serving — and a production inference fleet runs well above 28.6 %.

SDXL image generation — $ / 1,000 images @ US $0.15, 100 % utilization:

Card	All-in $/1,000 img	vs 3090-used	vs 3090-new
Arc B50	**$0.123**	—	—
Arc B70	**$0.141**	**33 % cheaper**	**47 % cheaper**
3090 (used)	**$0.211**	—	—
3090 (new)	**$0.265**	—	—

Table 14: All-in $/1,000 SDXL images, US $0.15/kWh, 100 % utilization.

SDXL is different in kind: the B70 wins on **both** the hardware-per-image term (it is *faster* per

image, §4.3) **and** the energy term (half the joules). When a card is cheaper on every component of the cost, there is **no crossover** —

> **Under our measured throughput/power and the stated TCO assumptions, SDXL is cheaper on Arc at every utilization and every electricity price.** In our model there is no operating point — no fleet, no country, no duty cycle — at which a 3090 generates SDXL images more cheaply than a B70 (let alone a B50). The B50 is the cheapest image-factory of all four parts.

Bottom line: the higher sticker on a B70 is repaid by the energy bill, not by marketing. On LLM serving it pays back above ∼29 % utilization (US) — i.e. on any real fleet — and on SDXL it is cheaper than a 3090 at every operating point in our TCO model.

5.2 Energy & carbon at fixed throughput

Flip the question: hold *output* constant and ask what it costs in power and carbon. This is the number a data-center operator or an ESG-minded investor cares about.

Fixed throughput	RTX 3090	Arc B70	Arc B50
1 M SDXL img / day	900 kWh/day, 328.5 MWh/yr	**450 kWh/day, 164.4 MWh/yr (−50 %, −66 t CO_2/yr)**	**359 kWh/day (−60 %)**
1 B tokens / day (LLM)	161 kWh/day	**111 kWh/day (−31 %)**	OOM (16 GB)

Table 15: Energy and carbon at fixed output. Arc cuts the energy bill 31 % (LLM) to 60 % (SDXL).

The diffusion number is the dramatic one: **a diffusion-heavy slice of an AI fleet halves its electricity and sheds ∼66 t CO_2/yr per million-image-per-day of capacity** simply by being Arc-class rather than 3090-class. Tie this to ColabHive's companion **6.1 TWh / 10-million-GPU** scaling vision: a diffusion-heavy slice of that fleet halves on Arc-class efficiency. As an *illustrative ceiling* — not a forecast — if the entire 6.1 TWh were SDXL-equivalent work running on 3090s, the same work on B70s would draw **3.05 TWh/yr**, a saving of **−3.05 TWh/yr ≈ the continuous output of a ∼350 MW power plant**.

Bottom line: at any fixed output, Arc cuts the energy bill 31 % (LLM) to 60 % (SDXL) and the carbon with it — and at fleet scale the diffusion savings alone are measured in power-plants, not percentages.

5.3 Rack / chassis density

Power, not slots, is the binding constraint in a modern rack. Sizing by **total board power (TBP — 3090 350 W / B70 230 W / B50 70 W)** into a fixed **10 kW rack** shows what you can actually deploy per kilowatt:

Per watt, the B50 packs **3.3× the VRAM and 2.2× the SDXL throughput** of a 3090; the B70 packs **2× the VRAM and 1.5× the LLM throughput**. Put the other way — **value-tier sizing to match one used 3090's SDXL throughput** takes only **0.89 of a B70 ($845, 205 W)** or **2.28 B50s ($796, 160 W, 36.5 GB)**. The B50 path matches the 3090's image throughput for

Per 10 kW rack	RTX 3090	Arc B70	Arc B50
Cards	28	**43**	**142**
Aggregate VRAM (GB)	672	**1,376 (2.05×)**	**2,272 (3.38×)**
SDXL throughput (img/min)	204	**353 (1.73×)**	**454 (2.23×)**
LLM throughput (tok/s)	16,296	**23,878 (1.47×)**	OOM (16 GB)
VRAM per kW (GB/kW)	68.6	139.1	**228.6**

Table 16: Rack / chassis density into a fixed 10 kW power budget, sized by total board power.

less capital, half the power, and more VRAM — and on a card with **no external power connector at all**.

Bottom line: the no-external-power 70 W B50 is the density play — in a power-bound rack it delivers 3.3× the VRAM and 2.2× the SDXL throughput per kilowatt of a 3090, and matches a used 3090's image output for less money and half the power.

Conservative note. All three subsections scale cards *linearly* and ignore the ∼10–20 % PSU/cooling overhead a real rack pays on top of card TBP. Because that overhead scales with card wattage, it penalizes the higher-wattage 3090 fleet *more* than the lower-wattage Arc fleet — so every density and energy figure here **understates** the Arc advantage rather than inflating it.

6 Honest ecosystem-maturity assessment

The hardware case is strong; intellectual honesty about the software stack is what makes it credible.

Where Intel XPU lags today:

- **4-bit QLoRA** initially appeared to fail, but the cause was a **SYCL device-selector mismatch** in bitsandbytes' 4-bit custom op under `ONEAPI_DEVICE_SELECTOR=level_zero:0`, *not* a missing Triton-XPU kernel; with `ONEAPI_DEVICE_SELECTOR=*:gpu` it trains end-to-end and beats the 3090 (§4.7). Both **LoRA-fp16 and 4-bit QLoRA** are now working Intel finetuning paths (§4.4, §4.7). The open item is wiring the selector into the platform's Intel training-launch path.
- **No FlashAttention / no xformers** on XPU (CUDA-only upstream); attention falls back to `torch_sdpa` / oneDNN paths.
- **Small-operator overhead** (TabNet, raw-eager training): the XPU under-saturates and per-kernel/eager cost dominates (§4.6).
- **Classical gradient boosting (XGBoost, CatBoost) has no Intel-GPU backend** — these stay on CPU on Intel (CUDA-only / experimental SYCL). Not a B70 deficiency; a library reality.
- vLLM-XPU has quantization gaps (no XPU INT8 scaled-mm kernel, §4.10) and no CUDA-graph equivalent; Triton-XPU is still out-of-tree.

- **Multi-card tensor-parallel (TP>1) is unvalidated on Arc**, and we observe a **real TP fault on dual-B70 in the current vLLM-XPU** (TP>1 does not run clean across two Arc cards in our environment). This is the single most important gap for the on-node 70 B / Battlematrix story (§2.3) — single-card efficiency is proven, multi-card composition is not.

The trajectory is fast and favorable:

- **Native** `torch.xpu` has been in PyTorch since 2.5 (Arc A-Series + Data Center Max named first), with **Arc B-Series (Battlemage) support maturing through 2.6–2.7**; the part we benchmarked runs `torch 2.10.0+xpu`.
- **bitsandbytes 0.48** added official Arc B-Series support (the 4-bit path is the immature piece, not bnb-on-Arc per se).
- **IPEX is folding into upstream PyTorch** (the standalone package reaches EOL ∼March 2026) — i.e. the "you need a special extension" era is ending.
- **Triton-XPU is being upstreamed** (relevant to other quantization kernels, though — per §4.7 — *not* the gating dependency for the bitsandbytes 4-bit QLoRA path, which already works on Battlemage with the correct SYCL device selector).

A concrete integration finding from this work: a training image built naively on `ubuntu:24.04` + a hand-installed compute-runtime **segfaulted** on `torch.xpu` device enumeration, despite a correct torch version. The fix was to build the training image on **Intel's own** `llm-scaler-vllm` **runtime base** (the same coherent oneAPI/runtime stack used for inference). The lesson — *use Intel's curated runtime, don't assemble your own* — is itself a useful data point for anyone standing up Arc training, and it confirmed the operator's hypothesis that "the key is in the llm-scaler."

7 Conclusion & roadmap

Measured on production hardware with device-attributed energy, **Intel Arc Pro (Battlemage) is a practical, power- and cost-efficient alternative to the RTX 3090 for the affordable-tier modern-AI workloads** — LLM serving, LoRA/QLoRA fine-tuning, and Stable Diffusion — delivering **95–112 % of the throughput at 56–64 % of the power (∼1.45–2.0× perf/watt)**, at **∼0.4× the $/GB of VRAM versus a new 3090 (∼0.6–0.8× versus a used one)**. It is genuinely *better* on diffusion and on QLoRA, ∼parity on LoRA and (slightly behind, 95–97 %) on LLM serving, and decisively better on perf/watt across the board. It is honestly weaker on small-operator workloads (TabNet — a loss that is, per the §4.7 note, *unaudited*) and has a defined, closing set of software-ecosystem gaps (FlashAttention, classical-ML-on-GPU). This is a **value-tier second-source case** — not a claim against frontier H100/B200 silicon.

Recommended deployment patterns (today):

- **LLM serving and SDXL inference on Arc Pro** — best perf/watt and economics; deploy first.
- **LoRA-fp16 and 4-bit QLoRA fine-tuning on Arc Pro** — both faster and more energy-efficient than the 3090; the production LLM-customization paths (QLoRA needs the `*:gpu` selector wired into the training launcher — a small integration task).

- **Keep deep-tabular/TabNet on NVIDIA** for now; classical GBDT stays on CPU on either platform.

Expansion path (per Intel's proposal to broaden the engagement): scale Arc Pro capacity for the inference + LoRA tier, add **Xeon** for the CPU/classical tier (where it is vendor-neutral anyway), and evaluate **Crescent Island** for the next-generation inference fleet as the XPU software stack closes its remaining gaps. The combination directly serves the original mission: **affordable, scalable, power-efficient AI compute for the LATAM developer and research community.**

8 Appendices — TCO model, raw data, reproduction & upstream report, methodology

Appendix A — The TCO model (formula, assumptions, sensitivity)

Everything in §5.1–5.3 is computed from this one model; a reviewer can recompute every cell from it.

All-in cost per unit of work. For a card producing throughput T units/hour at average power P watts:

```
all_in_$/unit(U) = capital / (T x hours_life x U)        <- hardware term
                 + (P / 1000 / T) x price_kWh            <- energy term

where:
  units_over_life     = T x hours_life x U
  energy_per_unit_kWh = (P watts / 1000) / (T units/hour)
```

Assumptions (identical to the §5 box): `hours_life = 26,280 h` (3-yr straight-line). `price_kWh` $\in$ {US 0.15, AR 0.10, DE 0.30}. Grid carbon `= 0.4 kg CO2/kWh`. Capital: `3090-used $875`, `3090-new $1,499`, `B70 $949`, `B50 $349`. T and P are the §4 measured cells (LLM serving C=16: 3090 582 tok/s @ 343 W, B70 555.3 tok/s @ 219 W; SDXL: 3090 7.3 img/min @ 394 W, B70 8.2 img/min @ 221 W, B50 3.2 img/min @ 69 W).

Crossover utilization. Setting `all_in_B70(U) = all_in_3090used(U)` and solving for U on the LLM cell (where the B70 wins on energy but not hardware) gives the closed form used in §5.1:

```
U* = dHW_constant / price_kWh = 0.0429 / price_kWh
  -> US 0.15 -> 28.6 %   |   DE 0.30 -> 14.3 %   |   AR 0.10 -> 43 %
```

For SDXL the B70 wins **both** terms, so the equation has no positive-U crossover $\rightarrow$ unconditional win (§5.1).

Sensitivity (the two knobs that move the answer):

- **$/kWh.** The energy term — and the *entire* B70 LLM-serving advantage — scales linearly with electricity price. Cheaper power (Argentina) pushes the LLM crossover up to 43 % util and shrinks the margin; expensive power (Germany) drops it to 14.3 % and widens it. SDXL is insensitive in sign (always wins) and only varies in *magnitude*.

- **Utilization *U*.** U only scales the hardware term, never the energy term. So every result here is **most favorable to the cheaper-energy card (Arc) at high utilization** and most favorable to the cheaper-sticker card (used 3090) at low utilization. A production fleet lives at high U, which is the regime where Arc wins.

Appendix B — Consolidated raw-data table (every measured cell)

So a reviewer can recompute J/unit, perf/watt, and the §5 economics independently. All cells measured 2026-06-21, bf16/eager, single dedicated GPU. "—" = not run / not applicable; "OOM" = did not fit in VRAM.

Workload (unit)	Metric	RTX 3090	Arc B70	Arc B50
LLM serving C=1 (tok/s)	throughput	37.8 ± 3.4	36.6 ± 0.1	—
LLM serving C=8 (tok/s)	throughput	296.7 ± 6.9	287.0 ± 0.5	—
LLM serving C=16 (tok/s)	throughput	582.0 ± 19.6	555.3 ± 2.1	OOM
LLM serving C=16	tok/J	1.723	2.511	—
LLM serving (load)	avg power (W)	∼343	∼219	—
SDXL (per image)	latency (s)	8.21 ± 0.09	7.33 ± 0.03	18.74
SDXL (per image)	throughput (img/min)	7.3	8.2	3.2
SDXL (per image)	energy/image (J)	3,240 ± 37	1,621 ± 6	1,291
SDXL	avg power (W)	∼394	∼221	∼69
LoRA-fp16 7B (step)	thr. (st/s, tok/s)	1.90, 1,944	1.96, 2,007	OOM
LoRA-fp16 7B	energy/step (J)	180.8	110.1	OOM
LoRA-fp16 7B	avg power (W)	∼347	∼222	OOM
QLoRA-NF4 7B (step)	thr. (st/s, tok/s)	0.889, 911	0.949, 972	0.373
QLoRA-NF4 7B	energy/step (J)	459.0	242.3	187
QLoRA-NF4 7B	avg power (W)	∼408	∼230	∼70
QLoRA-NF4 7B	VRAM 4-bit (GiB)	5.45	5.45	5.45
SDXL UNet-LoRA (step)	throughput (st/s)	1.625	0.930	—
SDXL UNet-LoRA	energy/step (J)	208	161	—
SDXL UNet-LoRA	avg power (W)	∼346	∼152	—
TabNet 16k×64 (rows/s)	throughput	16,852	7,739	3,700
TabNet	avg power (W)	∼108	∼87	∼39
TabNet	rows/J	159	89	—
Qwen2.5-14B bf16	weight load	OOM (24 GB)	loads (31.1 GiB)	—
AWQ-4bit C=1 (tok/s)	throughput	40.8	48.9	—
AWQ-4bit C=8 (tok/s)	throughput	302.9	381.8	—
AWQ-4bit C=16 (tok/s)	throughput	598.0	709.5	—

Table 17: Consolidated raw data, every measured cell.

Energy method per cell: Intel = Xe sysfs exact counter (all cells); NVIDIA = NVML exact counter for QLoRA, `nvidia-smi power.draw` @200 ms integral for LLM/SDXL (asymmetry flagged in Appendix D).

Appendix C — Reproduction & upstream report

C.1 Reproducing the headline cells. Pin a single dedicated GPU per side, drain other models off it, bf16/eager throughout, identical prompts/inputs/lengths. LLM throughput from vLLM's `vllm:generation_tokens_total` delta over a `MEASURE_START/END`-bracketed window (`ignore_eos` to fix output length). Energy from the exact accumulating counters where possible (Xe `energy1_input`, NVML `nvmlDeviceGetTotalEnergyConsumption`). Software versions in Appendix D. The QLoRA cell requires the selector fix below.

C.2 Ready-to-file bug report — bitsandbytes 4-bit QLoRA on Intel Arc B-series (device-selector trap).

Title: 4-bit QLoRA fails on Intel Arc B-series under `ONEAPI_DEVICE_SELECTOR=level_zero:N` — SYCL `No device of requested type available` (bnb's 4-bit op rejects the Level-Zero selector that Intel's own multi-GPU docs recommend)
Components: `bitsandbytes` (XPU 4-bit custom op) · Intel IPEX / `llm-scaler` multi-GPU documentation
Environment: Arc Pro B70 (`0xe223`, BMG-G31, 32 GB) and Arc Pro B50 (`0xe212`, 16 GB); `torch 2.10.0+xpu`; `bitsandbytes 0.49.2`; compute-runtime 26.05+; `xe` driver; Ubuntu 24.04 / kernel 6.17; image built on `intel/llm-scaler-vllm:0.14.0-b8.3.1` base.
Failure signature: With `ONEAPI_DEVICE_SELECTOR=level_zero:0` (per Intel's multi-GPU pinning guidance), loading an NF4-quantized model and starting a QLoRA step makes bitsandbytes' native 4-bit op (`torch.ops.bitsandbytes.quantize_4bit`) throw a SYCL `No device of requested type available`. Note `torch.xpu.is_available()` is `True` and `torch.xpu.get_device_properties()` enumerates the GPU correctly — only bitsandbytes' separately-compiled SYCL kernel queue rejects the device.
Root cause: bitsandbytes' 4-bit SYCL queue does not resolve a device when the process is scoped to the **Level-Zero** backend selector. `torch.xpu` and bnb's SYCL runtime resolve devices through different paths; the Level-Zero-only scoping that satisfies torch starves bnb's queue.
Fix (one line of configuration): Use the any-backend GPU selector and pin the specific card by affinity mask instead of by Level-Zero index:

```
# FAILS for bnb 4-bit:  ONEAPI_DEVICE_SELECTOR=level_zero:1
# WORKS:
export ONEAPI_DEVICE_SELECTOR='*:gpu'
export ZE_AFFINITY_MASK=1        # pin to the desired card (0-based)
```

With this, 4-bit QLoRA trains end-to-end (verified genuinely 4-bit: NF4 7B = 5.45 GiB).
Affected scope: the **entire Arc B-series** (reproduced on **both B70 and B50** with the identical fix). Critically, it affects **Project Battlematrix (8×B60)** and any multi-card Arc deployment, because per-card pinning is mandatory there and the standard `level_zero:N` guidance is exactly what triggers the failure — so following the standard multi-GPU docs is exactly what surfaces this failure.
Requested doc change (Intel): in the IPEX / llm-scaler multi-GPU pinning guidance, note that workloads using bitsandbytes 4-bit must pin with `ONEAPI_DEVICE_SELECTOR='*:gpu'` + `ZE_AFFINITY_MASK=<idx>`, not `level_zero:<idx>`.
Requested fix (bitsandbytes): make the 4-bit SYCL kernel queue resolve a device under the Level-Zero backend selector (or emit an actionable error pointing at the selector rather than the opaque `No device of requested type available`).

Appendix D — Methodology, versions, reproducibility (provenance)

Software versions (as benchmarked):

- Intel inference: `inference-ipex:v0.7.24` ← `intel/llm-scaler-vllm:0.14.0-b8.3.1` (`torch 2.10.0+xpu`, vLLM-XPU, compute-runtime 26.09).
- Intel training: `training-ipex:v0.2.0` (llm-scaler base + `peft`, `trl`, `bitsandbytes 0.49.2`, `pytorch-tabnet 4.1.0`; PIP_CONSTRAINT pins `torch==2.10.0+xpu`).
- NVIDIA: `inference-vllm:v2.0.8`, `inference-generative:v3.0.0`, `training-transformers-cu121:v1.0.5`, `training-pytorch-cu121:v1.0.1` (TabNet baseline required runtime `numpy<2` + `pytorch-tabnet==4.1.0` due to a numpy-2/torch-2.1.2 incompatibility in the older image).
- Host: Ubuntu 24.04, kernel 6.17, `xe` driver, compute-runtime 26.05+, Resizable BAR on.

Energy method (and its asymmetry, stated plainly): Intel = `xe` sysfs `energy1_input` (μJ, exact accumulating counter, per-card by PCI id). NVIDIA = the exact NVML `nvmlDeviceGetTotalEnergyConsumption` counter where used (QLoRA), else the integral of `nvidia-smi power.draw` @200 ms (LLM, SDXL). **The two NVIDIA methods are not the same instrument** — 200 ms sampling can miss sub-sample transients — so the LLM/SDXL perf/watt margins carry a small extra uncertainty the QLoRA cell (exact counter, both sides) does not. Window bracketed by workload-emitted `MEASURE_START/END`. Now collected fleet-wide by node-runtime 0.10.119 → `node_power_samples`.

Workloads: Qwen2.5-7B-Instruct (LLM, vLLM and LoRA/QLoRA); Stable Diffusion XL base 1.0 (inference + UNet-LoRA training); synthetic tabular (TabNet). bf16/eager throughout; identical prompts/inputs/lengths cross-vendor.

Data provenance / open items:

- **K=3 (±1 sample-std, n=3)** covers the two inference cells — LLM serving (§4.2) and SDXL (§4.3). LoRA (§4.4) and QLoRA (§4.7) are paired measured runs reported as **point estimates** (repeated, no tight-CI claim). SDXL-UNet-LoRA (§4.5) and TabNet (§4.6) are single runs. **Sampling caveat:** all "K=3" is *same-device* repetition — **N=1 hardware per vendor** — so silicon-lottery / board-partner / thermal variation is uncontrolled; a second physical pair and a second model size (∼1.5 B) are the next steps. The 14 B VRAM-headroom case (§4.8) is measured.
- **Known confounds (not yet eliminated):** (a) *engine-version skew* — Intel runs vLLM-XPU 0.14.x on `torch 2.10+xpu`; NVIDIA runs stock vLLM (`inference-vllm:v2.0.8`, CUDA cu121 lineage). bf16/eager is pinned for fair fast-paths, but they are different vLLM forks/versions, so near-parity reflects silicon *plus* engine, not silicon alone. (b) *host-CPU mismatch* — Intel host i7-10700 (8c) vs NVIDIA Xeon E5-2680 v4 (28c); irrelevant for GPU-resident kernels but a live confound for the host-sync-bound **TabNet** cell specifically (its 2.2× loss may be partly host-CPU, and is unaudited).
- **QLoRA-4bit now runs on the B70** (§4.7) — 0.949 steps/s · 242.3 J/step, ∼7 % faster and 1.89× more efficient than the 3090, verified genuinely 4-bit (5.45 GiB on both vendors). The earlier "fails" result was a **SYCL device-selector mismatch** (`level_zero:0` → bnb's 4-bit op throws "No device"; `*:gpu` fixes it), **not** a Triton-XPU gap as first hypothesized.

- Intel **does not publish dense BF16 TFLOPS** for Arc Pro B-series; the compute-economics table uses published INT8 TOPS only.
- A separate platform improvement was identified (not yet shipped): the node runtime reads Intel VRAM via `xpu-smi` (which reports "No device" on the B70) and falls back to accounting; reading `torch.xpu.mem_get_info()` would give live, accurate Intel VRAM and improve placement.
- Market-share, pricing, CUDA-EULA, and spec figures in §2 and §5 are from public sources (TechInsights/HPCwire, NVIDIA CUDA EULA, Intel datasheets/newsroom, vendor spec pages); street prices are volatile and flagged as such.

Appendix E — Sources (public)

External market, pricing, licensing, and hardware-spec figures in §2 and §5 are from public sources; street prices and shipping specs are volatile and were current at the 2026-06-21 measurement date.

1. NVIDIA data-center GPU market share (∼98%, 3.76M of 3.85M units, 2023) — TechInsights, reported via HPCwire.
2. NVIDIA CUDA End User License Agreement, §1.2 "Limitations" (item 8, on translating CUDA output to non-NVIDIA platforms) — NVIDIA.
3. ZLUDA (CUDA-on-non-NVIDIA) project takedown at AMD's request, August 2024 — project repository and press coverage.
4. Blackwell supply ("sold out ∼12 months ahead," Oct 2024) and per-GPU pricing ($30,000–40,000, Jensen Huang) — NVIDIA management remarks and press.
5. Intel Arc Pro B-series (Battlemage), Project Battlematrix (8×B60 → 192 GB), and the llm-scaler software stack — Intel datasheets and newsroom.
6. bitsandbytes Arc / XPU 4-bit support — bitsandbytes release notes.
7. Native torch.xpu support timeline (PyTorch 2.5 onward; Battlemage maturing 2.6–2.7) and IPEX standalone EOL (∼March 2026) — PyTorch and Intel documentation.
8. GPU specifications (VRAM, memory bandwidth, TBP, INT8 TOPS) for the RTX 3090 and Arc Pro B70/B60/B50 — vendor spec pages.