Exact Continuation Across CPU, Apple GPU, and NVIDIA GPU

VertRule technical note

2026-10-09 · v1 · Qwen3-0.6B · Apple M4 Pro CPU + GPU · NVIDIA Tesla T4

1. Claim

A single language-model generation was paused mid-answer on a CPU, resumed on an Apple GPU, paused again, and finished on an NVIDIA GPU in a different machine. At every position, the full execution state matched an uninterrupted CPU run of the same computation bit for bit, and so did every generated token and the terminal state. All 34 acceptance checks passed.

In a second, separate experiment, the process producing a generation was killed with `SIGKILL` after publishing a checkpoint, and a fresh process on a different backend resumed from it. This was done in both directions between a CPU path and a hybrid CPU + Apple GPU path. In both directions the final saved state was byte-identical to an uninterrupted reference: 458,752 values, none different.

This is a systems-verification result. It is not a benchmark, it makes no performance claim, and it does not claim that the backends involved are equivalent in general (Section 8).

2. Why this question

Inference state already moves between processes and machines. Serving systems hand key/value caches between prefill and decode workers, persist them to shared storage, and reuse them across requests. Long-running jobs are restarted after failure. The evidence that a move was correct is usually indirect: the output still looks reasonable, or downstream accuracy holds.

Indirect evidence is the norm because different hardware ordinarily computes slightly different numbers. Vendors choose different kernels, reduction orders, and fused instructions, and floating-point arithmetic is not associative. Recent work has made inference reproducible run to run on one GPU type, by making kernels batch-invariant [1], and across generations of one GPU vendor, for linear layers, by fixing the reduction order [2]. Formal treatments now define deterministic inference as identical logits across executions [3].

This note asks a narrower and stricter question across a wider hardware boundary. Given state produced on one kind of hardware, does continuing on another kind of hardware reproduce the *same computation*, not merely an acceptable one? The check is direct: compare the complete internal state at every position against a run that never moved.

3. Setup

Model and policy. Qwen3-0.6B (dense, 28 layers), 32-bit floating-point execution, greedy decoding. Prompt: “The capital of France is”.

Hardware.

- Apple M4 Pro, CPU path.
- Apple M4 Pro, Metal GPU path, on the same machine.
- NVIDIA Tesla T4 (compute capability 7.5) on a rented cloud host running Ubuntu 24.04, with GPU programs compiled at run time. A GeForce GTX 1050 was used as a rehearsal host.

What "exact" means here. Equality is checked on three surfaces, all with no tolerance:

- *State.* A cryptographic digest over the sequence position and every key and value in the attention cache for all 28 layers. Per-layer digests are recorded as well, so any disagreement can be localized.
- *Output scores.* A digest over the full row of next-token scores (the logits) at each step.
- *Tokens.* The selected token at every step, and the whole generated sequence.

Discipline. For both experiments, the acceptance predicate was written down and committed before any run, and the reference owns every expected value. In the first experiment, the reference was frozen before any migration segment ran. Predictions made during design were treated as predictions only.

One arithmetic specification. All backends execute a single declared arithmetic specification for this model. A backend is allowed to run a step only if it realizes that specification; a backend that cannot is refused before it touches the model. The checkpoint records the logical state of the computation, not which hardware produced it. This is what makes an exact cross-hardware handoff a well-posed question rather than a hope.

4. Experiment A: CPU → Apple GPU → NVIDIA GPU

Protocol

1. Reference. Run the six-token generation uninterrupted on the CPU path. Freeze its state digest after every token, its output-score digests, and its tokens.
2. Cross-check. Run the same generation uninterrupted on the Apple GPU. It must agree with the reference at every step. Had the two disagreed anywhere, the experiment would have halted. The reference is never chosen to match the chain.
3. Segment 1 (CPU). A fresh CPU process generates two tokens and publishes checkpoint C_a at position 6.
4. Segment 2 (Apple GPU). A new process verifies C_a, restores it on the Metal GPU, generates two more tokens, and publishes checkpoint C_b at position 8.
5. Segment 3 (NVIDIA GPU). C_b, the model weights, and a run script are shipped to the T4 host. The checkpoint is verified there, restored on the GPU, and the final two tokens are generated. Each GPU step is checked against a CPU oracle.
6. Composition. An assembler derives the verdict from the segment records alone. Each record is bound by its digest, and no state digest or token is entered by hand.

Results

Table 1. State at each checkpoint and step, compared to the uninterrupted reference.

Position	Produced by	Compared against	State digest	Equal
6	CPU segment, then restored on Apple GPU	Reference; checkpoint manifest and envelope	`6a3934bf…`	yes
8	Apple GPU segment, then restored on NVIDIA GPU	Reference; checkpoint; all 56 per-layer K/V digests	`d715eaf3…`	yes
9	NVIDIA GPU step	Reference; CPU oracle	`299ffafa…`	yes

Position	Produced by	Compared against	State digest	Equal
10	NVIDIA GPU step (terminal)	Reference; CPU oracle	`8e35fcc0…`	yes

Table 2. Generated tokens.

	Tokens 1–2	Tokens 3–4	Tokens 5–6
Hardware	CPU	Apple GPU	NVIDIA GPU
Token ids	12095, 13	576, 6722	315, 15344
Text	” Paris.”	” The capital”	” of Italy”
Equal to reference	yes	yes	yes

The Apple GPU segment also matched the reference's output-score digest at every step it executed. On the NVIDIA GPU, every arithmetic operation of both steps ran on the GPU, and none fell back to the host processor.

The generated text is beside the point. A 0.6-billion-parameter model continuing “The capital of France is” with ” Paris. The capital of Italy” is ordinary model behaviour. What matters is that the identical continuation, down to the last bit of internal state, came out of three different kinds of hardware.

Composition receipt. PASS, 34 of 34 checks. Two negative checks show that assembly fails when it should:

- A chain record with a tampered final state fails the terminal-state check.
- Substituting the uninterrupted Apple GPU run for the resumed Apple GPU segment fails three restoration checks.

What differed, and did not matter. The GPU programs compiled on the T4 are not reproducible. Two T4 runs differed in the compiled-program digests of two kernels. The T4 run and the GTX 1050 rehearsal differed in exactly 72 recorded fields, all describing the compilation environment: GPU model, compute capability, compiler version, and compiled program. Every state, score, and token was identical across all of them.

Provenance check. The command-line binary used for the first runs was built from a working tree with unrelated uncommitted edits. To bind the result to committed source, the binary was rebuilt from the committed revision and the CPU and Apple GPU runs were repeated. The evidence records and both checkpoints' payloads came out byte-identical.

5. Experiment B: crash and resume on the other backend

Protocol

Each act runs as its own operating-system process, and all use the same release binary.

1. Reference. An uninterrupted CPU run generates four tokens and publishes a final checkpoint.
2. Producer. A process on backend X generates two tokens and publishes a checkpoint. Once its readiness event is received, the input prompt file is deleted and the process is killed with `SIGKILL` while still running.
3. Consumer. A fresh process on backend Y verifies the checkpoint, restores it, generates the remaining two tokens, and publishes its own final checkpoint.
4. Negative controls. Five fresh processes attempt invalid resumes (Table 4).

5. Verification. A separate process recomputes every record digest and compares the two final checkpoints bit for bit. It loads no model, opens no GPU, and trusts no producer assertion.

The two directions are X = CPU, Y = hybrid, and X = hybrid, Y = CPU. In the hybrid path, the large matrix multiplications execute on the Apple GPU and the rest of the computation runs on the CPU.

Results

Table 3. Acceptance, both directions.

Check	CPU → hybrid	hybrid → CPU
Producer alive at kill, terminated by signal 9	yes	yes
Consumer is a distinct, fresh process	yes	yes
Checkpoint verified before restore	yes	yes
GPU work in hybrid process / fallbacks	394 dispatches / 0	1,182 dispatches / 0
Output-score digest equal at each resumed step	yes	yes
Selected tokens equal	yes	yes
State digest equal at each resumed step	yes	yes
Final checkpoint vs reference	bit-exact, 458,752 values, 0 differ	bit-exact, 458,752 values, 0 differ
No duplicated or skipped generation step	yes	yes
Independent verifier verdict	verified	verified

Both directions generated ” Paris. The capital”. The checkpoint published by the hybrid producer was byte-identical to the one published by the CPU producer at the same position. The two backends did not just converge on the same answer; they wrote the same bytes.

Table 4. Negative controls, both directions. Every invalid resume was refused before producing output.

Control	Outcome
One bit flipped in the checkpoint payload	refused (checkpoint integrity)
Checkpoint bound to a different program identity	refused (binding mismatch)
Checkpoint missing	refused
Resume requested on an unsupported full-GPU path	refused before any model or device was loaded
Fresh run requested on the same unsupported path	refused before any model or device was loaded

The verifier was also shown to reject tampering with the evidence itself. Altering one recorded output-score digest, or flipping one field of the comparison record, makes verification fail.

Clean-checkout reproduction. From a fresh clone with no untracked files and no producer processes, the verifier was rebuilt and rerun for both directions. Both verdicts and both receipt digests were reproduced.

The only field that changed was the verifier's own process id. The rebuilt verifier binary had the same SHA-256 as the original.

6. What this demonstrates

- Execution state can outlive the hardware that produced it, exactly. A checkpoint written on one kind of processor is continued on another with no change in any bit of state, score, or token.
- The check is direct, not inferred. Fidelity is established by comparing full state against an uninterrupted run at every position, not by inspecting output quality.
- Failure is crash-shaped, not polite. The producer is killed while still running, with no chance to clean up, and the resume still matches exactly.
- Invalid resumes fail closed. Corrupted, mis-bound, missing, or unsupported inputs produce a refusal, never a plausible-looking continuation.
- Evidence is checkable without trust. Verdicts are derived from digest-bound records by a separate process. Tampering with either the state or the evidence is detected.

7. Related work

Run-to-run determinism. Batch-invariant kernels remove the main source of nondeterminism in GPU serving, which is dependence of reduction order on batch size, and yield identical outputs across repeated runs on one GPU type [1].

Cross-architecture determinism on one vendor. Fixed-configuration kernels whose reduction order depends only on problem shape give bitwise-identical linear-layer outputs across several NVIDIA GPU generations [2]. A formal specification of deterministic inference as identical logits across executions, with an analyzer for GPU kernels, is given in [3].

State transport. Key/value-cache transfer, persistence, and reuse are established practice in disaggregated and cached serving. Their correctness rests on exact prefix reuse or on downstream accuracy, not on a direct comparison of continued state.

This note differs in two ways. The boundary crossed is between processor families (CPU, Apple GPU, and NVIDIA GPU) in the middle of a single generation. And the object compared is the full continued execution state at every position, not only the final output.

8. Boundary

This note does not claim:

- That any two backends are equivalent in general. The result holds for one model, one prompt, greedy decoding, and the declared arithmetic specification.
- Behaviour on other models, other precisions, long contexts, sampling, or batched serving.
- That the NVIDIA GPU path is qualified for production use. It ran as a verified segment, outside the production generation session.
- Reproducible GPU program compilation. It was not reproducible, and only compilation fingerprints differed.
- An unbroken NVIDIA GPU checkpoint chain. The NVIDIA segment does not yet publish its own production checkpoint.
- A full-GPU path on Apple hardware for Experiment B. The resumable GPU path there is hybrid, and full-GPU resume was refused by design.
- Anything about speed. Performance was not an acceptance criterion and was not measured.
- Anything about what the model's state means. This is a statement about bits, not about cognition.

9. Evidence identities

All digests are BLAKE3 unless marked otherwise.

Experiment A composition receipt (34/34):
`e35fad84947f9a72bd3ba6c864ed364325bd8e0cb6f42889954e71ead92c74c3`

Experiment A uninterrupted reference:
`1eb9e0270722207af476fa3d20d41dc68858720338850bc32dd8f870dd5b994d`

Experiment A NVIDIA GPU segment:
`4988b46af2f0035910ff58e0b3fd0ec1cac0501dc5a5fdb0cd2f1d4e6f2c4934`

Experiment B receipt, CPU $\rightarrow$ hybrid:
`cd93584028ae2e9c058a803ac3dc622d53646baa2a7fa40e7e5bd50618eb4023`

Experiment B receipt, hybrid $\rightarrow$ CPU:
`864823715873acef882a030e4b0e5ed2d087b5e87ea520464eb85d708b7e704b`

Experiment B verifier binary (SHA-256):
`95faa802c49fed664a9dc3f0ffdf3db08cee365d4e5825a4d75e56c71631d6e8`

References

[1] H. He and Thinking Machines Lab. "Defeating Nondeterminism in LLM Inference." *Thinking Machines Lab: Connectionism*, September 2025.

[2] "Accelerating the Mitigation of LLM Inference Nondeterminism Across GPU Architectures." arXiv:2609.25624, 2026.

[3] "Vosti: Specifying, Implementing, and Verifying Deterministic LLM Inference." arXiv:2609.38981, 2026.