ANE Acceleration for llama.cpp — ggml CoreML Backend + QVAC Integration

Title	ANE Acceleration for llama.cpp — ggml CoreML Backend + QVAC Integration
Publish Date	Apr 27, 2026
Submission End Date	Jun 1, 2026
Reward	5'000 USDt

Problem Summary

What is the problem, deficiency, or gap which will be addressed by this project?

The Apple Neural Engine (ANE) shipped on M3 / M4 / M5 Macs and A17 Pro / A18 / A19 iPhones and iPads delivers high-throughput int8 matrix multiplication at a fraction of the energy cost of the Metal GPU path. llama.cpp and ggml currently route all LLM inference through Metal or CPU, leaving the ANE unused on every Apple device that ships one.

Draw Things recently demonstrated that ANE can be integrated as a **targeted accelerator for int8 matmul** inside a custom inference stack — using CoreML exclusively as a front door to ANE — while the host runtime keeps full ownership of intermediate allocations, KV cache, kernel caching, and scheduling. This grant funds the integration of that architectural pattern into llama.cpp / ggml and the QVAC addon stack: ANE accelerates matmul, Metal / CPU continue to own everything else, and the existing Metal path remains the fallback.

Impact / Reason Why

What is the metric or goal this bounty aims to improve?

- **Accessibility** — ANE is present on every current Apple Silicon device and entirely unused today by ggml-based LLM stacks; this grant unlocks it.
- **Performance** — Int8 matmul throughput on the ANE exceeds the Metal GPU path for prefill / prompt processing on long contexts, where invocation overhead is amortized.
- **Energy** — ANE is 3–5× more energy-efficient than the GPU for sustained LLM workloads; critical for iOS battery life and macOS laptop thermal budget.
- **Ecosystem growth** — Upstream contribution to llama.cpp / ggml benefits the wider community; mirrors the Draw Things architectural pattern rather than forking it.

Scope Items

What important items are in scope for the project?

- **Two PRs** — All work is delivered as two pull requests:
  - **PR 1** → ggml-org/llama.cpp (or the equivalent `ggml-org/ggml` staging repo) — a new CoreML/ANE acceleration path in ggml that offloads int8 matmul to the ANE while the rest of the graph (attention glue, norms, activations, sampling, KV cache) continues to run on Metal / CPU.
  - **PR 2** → tetherto/qvac-lib-infer-llamacpp-llm and tetherto/qvac-lib-infer-llamacpp-embed — integration of the accepted ANE path into the QVAC addons, with user-facing controls.
- **Public-API only** — Rely exclusively on public CoreML and `coremltools` APIs; no dependency on the private ANE framework and no baked-in weights.
- **Dynamic matmul specialization** — Compile and cache `mlmodelc` programs per unique `(M, K, N)` shape; reuse across layers, requests, and model instances; persist the cache to disk, versioned per model and per OS.
- **Direct int8 handoff** — Use macOS 26 / iOS 26 `MLMultiArray` int8 support to avoid fp16 / int32 promotion at the CoreML boundary and reduce cross-process traffic.
- **IOSurface-backed transfer** — Wrap ggml device buffers as `IOSurface`-backed `MLMultiArray`s for zero / low-copy handoff between ggml tensors and CoreML inputs / outputs.
- **Quantization bridging** —
  - **Phase 1** — `Q8_0` and `Q8_K` mapped to ANE-native int8 with row-wise / block-wise scales applied on the residual (post-matmul) path — the analogue of the "8-bit S" format used by Draw Things.
  - **Phase 2** — `Q4_0` / `Q4_K_M` / `Q5_K_M` / `Q6_K` via on-the-fly repack to int8 for ANE-eligible ops, with a documented memory / latency trade-off and optional pre-repack at model load.
- **Scheduling policy** —
  - **Prefill** — ANE-first for attention and MLP matmuls above a shape threshold that amortizes CoreML invocation overhead.
  - **Decode** — Metal-first by default; ANE opt-in for batched decode, speculative decoding, and long-context prefill where batches are large enough.
  - **Energy-first mode** — Allow ANE to be preferred even when absolute latency is slightly worse than Metal, for sustained workloads and mobile thermal budgets (M5 / A18+ class behaviour).
- **Transparent fallback** — Automatic fallback to Metal / CPU when a shape is not profitable, when the OS version is insufficient, when the device has no suitable ANE generation, or when thermal / power state requires it.
- **Mobile path** — Full support for iOS 26+ with the same acceleration surface as macOS, including background-thermal-aware throttling and correct behaviour under OS app-lifecycle transitions.
- **Model compatibility** — Validate on representative LLM families used by QVAC: Qwen 3.5 / 3, Gemma 4.
- **Target OS / hardware** —
  - macOS 26+ on M3, M4, M5.

    - iOS 26+ on A17 Pro, A18, A19 (and later).
    - Graceful no-op on older OS or pre-M3 / pre-A17 hardware (Metal path remains default).
- **Docs** — Build instructions, architecture overview, shape-cache semantics, quantization bridging notes, QVAC integration guide.
- **Tests** — Numerical parity vs the Metal baseline (per-layer + end-to-end sampling), cross-device CI (macOS on M-class, iOS on A-class), shape-cache invalidation tests.
- **Benchmarks** — End-to-end prefill / decode latency, tokens/sec, TTFT, peak memory, energy-per-token, sustained thermal behaviour. ANE + Metal hybrid vs Metal-only baseline on identical hardware and quantization.

Scope Exclusions

What important items are out of scope for the project?

- Private ANE framework usage (e.g. `AppleNeuralEngine.framework`); this grant is strictly CoreML-public-API based.
- Full replacement of the Metal backend — ANE accelerates selected ops only; Metal continues to own the graph.
- Training or fine-tuning on ANE.
- Pre-M3 / pre-A17 hardware (Metal path retained as default, unchanged).
- CUDA / ROCm / SYCL backends — out of scope and untouched by this work.
- GUI or developer tooling beyond a minimal benchmark harness.

Deliverables

What are the outputs that must be submitted?

- **PR 1 — llama.cpp / ggml** — New `ggml-coreml` backend module: CoreML-backed int8 matmul kernels, shape-keyed `mlmodelc` program cache with on-device first-use compile + persistent disk cache, IOSurface-backed `MLMultiArray` bridge, `Q8_0` / `Q8_K` native path, `Q4_*` / `Q5_K` / `Q6_K` repack path, scheduling hooks, Metal fallback, CI. Must follow llama.cpp / ggml conventions.
- **PR 2 — QVAC addons** — Integration of the ANE path into `qvac-lib-infer-llamacpp-llm` and `qvac-lib-infer-llamacpp-embed`, user-facing latency-first vs energy-first mode, per-device defaults, addon API docs. It must be supported as a dynamic backend on llama.cpp. You should be able to dl_open lib-coreml
- **Benchmark report** — Prefill / decode latency, tokens/sec, TTFT, peak memory, energy-per-token, sustained thermal behaviour. Platform matrix across M3 / M4 / M5 and A17 Pro / A18 / A19, quantization matrix across `F16` / `Q8_0` / `Q8_K` / `Q4_K_M` / `Q5_K_M` / `Q6_K`.
- **Cached `mlmodelc` artifacts** — For a representative subset of target shapes, published alongside the release so users start warm on first run.
- **Docs** — Build instructions, architecture notes, quantization guidance, addon API reference, thermal / energy tuning guide.

Acceptance Criteria / Definition of done

What needs to be achieved for this project to be considered completed?

- [ ] No measurable regression vs the Metal GPU baseline on any supported model / quantization / device, at equivalent quality.
- [ ] Prefill latency improved vs the Metal baseline by at least **1.5×** on M3 and M4 at `Q8_0` for long contexts (≥ 4 K tokens) *(⚠️ TBD — applicants should document methodology so the threshold can be re-evaluated)*.
- [ ] Energy-per-token improved vs the Metal baseline by at least **2×** on M3 / M4 / A18 under sustained generation, at equivalent quality *(⚠️ TBD)*.
- [ ] Numerical parity vs Metal within agreed tolerance (per-layer numerical delta analysis + end-to-end output equivalence sampling).
- [ ] Transparent fallback demonstrated on pre-M3 Macs and pre-A17 iOS devices, and on older OS versions — same code path, same binary, no user intervention.
- [ ] Clean compilation (`-Wall -Wextra`) on Clang across all target OSes.
- [ ] Shape-cache correctness under model-swap, quant-swap, and cross-process scenarios.
- [ ] CI green across macOS 26+ on M-class Macs and iOS 26+ on A-class devices.
- [ ] Both PRs follow their target repository's conventions and require only minor modifications before merge.
- [ ] A reviewer can build from source and run a benchmark on their own Apple Silicon device within 15 minutes using the README.

Milestones

What are the intermediary checkpoints necessary to track project progress?

ID	Description	Deliverables	Reward
M1	Architecture, prototype, capability assessment	• Architecture design; ANE / CoreML capability assessment across M3 / M4 / M5 and A17 Pro / A18 / A19; int8 matmul prototype running on ANE via CoreML; IOSurface-backed `MLMultiArray` proof-of-concept; shape-cache skeleton; benchmark harness. • **Delivers** — Design doc, working prototype, capability matrix,	1’500 USDt

		benchmark-harness scaffolding, CI skeleton green on macOS M3.	
M2	Core Q8 path end-to-end	• Full `ggml-coreml` backend for `Q8_0` and `Q8_K`: shape-specialized `mlmodelc` compile + cache, residual-path scale application, scheduling hooks (prefill / decode), Metal fallback, numerical parity validation. • **Delivers** — CLI / benchmark runs end-to-end on Q8 with ANE offload on macOS M3 / M4. Numerical parity within tolerance vs Metal. CI green on M-class.	1’500 USDt
M3	Low-bit path and mobile	• `Q4_0` / `Q4_K_M` / `Q5_K_M` / `Q6_K` on-the-fly repack path; iOS 26+ support on A17 Pro / A18 / A19; energy-first mode; thermal-aware fallback; extended model matrix (Llama / Qwen / Mistral / Gemma / Phi). • **Delivers** — All target quantizations running on ANE where profitable, on both macOS and iOS. Energy-first mode switchable at runtime. CI green on iOS.	1’500 USDt

M4	QVAC integration, full benchmarks, docs	• PR 2 to the QVAC addons; full benchmark campaign across model × quantization × device matrix; energy / thermal sweeps; documentation; final polish. • **Delivers** — Both PRs ready for review. QVAC addons expose latency-first vs energy-first controls. Benchmark report published. Docs complete.	1’500 USDt

Success Indicators / Key Results

What are the key results that help us measure if this project was a success?

- ☐ Build-to-first-token in under 15 minutes from the README on any supported Apple Silicon device.
- ☐ Prefill on a 4 K prompt completes ≥ 1.5× faster on M3 / M4 with ANE than with the Metal baseline at equivalent quality
- ☐ Energy-per-token on iOS 26+ (A18) ≥ 2× lower with ANE than with Metal under sustained generation.
- ☐ No measurable quality regression on a shared LLM evaluation harness (e.g. `lm-evaluation-harness` standard tasks) vs the `F16` / `Q8_0` Metal baselines.
- ☐ Two independent reviewers confirm output equivalence with the Metal baseline on the target model matrix.
- ☐ QVAC addons integration.

Applicants Requirements

What requirements that applicants must meet?

- C / C++ systems programming and CMake experience.
- Deep familiarity with ggml / llama.cpp internals (prior contributions strongly preferred).
- Working knowledge of Apple's CoreML, `coremltools`, IOSurface, Metal, and Objective-C++ / Swift interop.
- Experience with Apple Neural Engine characteristics: invocation overhead, tile / shape specialization, quantization semantics, int8 scale layouts.
- Access to Apple Silicon hardware across generations — M3, M4, plus at least one of M5 and A17–A19 class devices for end-to-end validation.

- Weekly progress updates in English via GitHub issues / PRs.

Reward & Payment Schedule

What are the payout structure and conditions?

Total: 5’000 USDt, four milestone-based installments:

- **M1** — 1’500 USDt
- **M2** — 1’500 USDt
- **M3** — 1’500 USDt
- **M3** — 500 USDt

Each milestone is reviewed within 5 business days. Payment released after reviewer approval. No partial payouts.

Resources & Links

What are additional useful links and resources for applicants?

- ggml-org/llama.cpp — Target repo for PR 1.
- ggml-org/ggml — Tensor library / backend host.
- tetherto/qvac-lib-infer-llamacpp-llm — Target addon for PR 2 (LLM).
- tetherto/qvac-lib-infer-llamacpp-embed — Target addon for PR 2 (embeddings).
- Draw Things — Making Apple Neural Engine Work in a Custom Inference Stack — Architecture reference.
- Apple CoreML — Public API used to dispatch to ANE.
- Apple coremltools — Offline compilation tooling.
- Apple IOSurface — Shared-memory surface for zero-copy tensor handoff.
- Apple Metal Performance Shaders — Comparison baseline.
- GGUF Specification — File format docs.