AIENOS & OMEGA · HARDWARE-VERIFIED TECHNICAL WHITEPAPER · APACHE-2.0

Playing Unplayable Games: Zero-CUDA Full-APU Co-Execution, Weight-to-Code Partial-Evaluating JIT, Exact Merkle Delta-Matmul, and Trap-Exact ARM64 NEON SDOT in Pure Seeded Omega

Author: Drake Stapleton · **Project:** AienOS & Omega (`omega-mv` , `omega-observer` , `omega-play`)
Date: 2026-11-26 · **License:** Apache-2.0
Canonical Repositories:

- `omega-observer` (`d079300`): `https://interplane.io/drakestapleton/omega-observer` · `https://github.com/drakestapleton/omega-observer`
- `omega-play` (`7dba48c5`): `https://github.com/drakestapleton/omega-play`
- `omega-mv` (`feat/a6-svec-sdot` @ `1da54a5de`): `https://interplane.io/drakestapleton/omega-mv`
- Live Workbench: `https://www.aienos.com/aien/` · Press Room: `https://www.aienos.com/press/`

> **Sovereign Intelligence™ Principle:** *Sovereign, not super — intelligence you own, run on hardware you own, and can check yourself.* Every quantitative claim in this whitepaper is backed by immutable, machine-readable hardware receipts (`MEASURED_DGX_SPARK_GRACE_A64` , `MEASURED_DGX_SPARK_GB10` , `MEASURED_MACBOOK_M2_ARM64`), pinned compiler SHA-256 digests, and an explicit honest-caveats section documenting every measured regression and scope boundary.

Abstract

Conventional AI and systems stacks treat software as a tower of opaque layers: Python orchestrators over C++ runtimes over proprietary GPU drivers (`nvidia.ko` , CUDA) and dynamic memory allocators. When pure seeded Omega (`seed/out/omc` , rebuilt deterministically from a 408-byte hand-written loader with zero C on its bootstrap path) was first benchmarked head-to-head against hand-tuned ARM NEON and LLVM Rust on dense CPU loops, scalar `omc` trailed hand-written `vdotq_s32` by **84.7×** on `Q8_dot` and trailed LLVM Rust `UOR-Foundation/uor-matmul` by **1.22×–7.40×**.

We attacked that gap simultaneously from two directions:

1. **Closing the native silicon compiler gap directly:** Landing 17 new ARM64 NEON/DotProd encodings, 128-bit integer vectors with `ADDV` reductions, and trap-exact 64-byte/iteration `Q8_dot` `SDOT` loop vectorization directly in `seed/ng/src/a64.om` (`feat/a6-svec-sdot` @ `1da54a5de`). On NVIDIA DGX Spark Grace Cortex-X925, `Q8_dot` dropped from **`3,055.13 ns`** to **`35.93 ns`** at `n = 4096` (**`85.03×`** speedup, within `1.073×` of unchecked `gcc -O3` `vdotq_s32`) and from **`12,126.79 ns`** to **`137.23 ns`** at `n = 16384` (**`88.37×`** speedup, **beating** unchecked `gcc -O3 -march=armv8.6-a+dotprod` median `137.68 ns` at `0.9967×`) while preserving exact bounds, fuel, and overflow traps.
2. **Playing architectural games standard stacks cannot play:**
   - **Game #1 — Weight-to-Code Partial-Evaluating `E8_I8` JIT (`om/e8_jit_gen.om`):** Compiles quantized lattice weights directly into straight-line pure-OM-2 bit-shifts (`<< 4` , `<< 5`) and additions with **zero weight RAM** and **zero multiplies** in compute, erasing `36.1%–84.7%` of weights at compile time. Wins **10/10** shapes on DGX Spark (**`1.54×–7.73×`** dense; **`11.38×–21.44×`** at `75%` zero codes, `6.75 us` vs `144.77 us` vs LLVM Rust `uor-matmul` , `159` -call break-even) and **9/10** shapes on Apple M2 (**`1.35×–7.22×`**).
   - **Game #2 — Algebraic 4×-Unrolled `E8_I8` & Exact Merkle Delta-Matmul (`om/omega_uor_worker.om` , `om/omega_uor_x925.om`):** Single-core pure-OM-2 beats `--release -C target-cpu=native` LLVM Rust `uor-matmul` even **stateless** (**`2.51×`** on `8×512×2048`) and hits **`2.95×`** / **`14.85 GMAC/s`** cached. Exploiting exact `i64` integer linearity (`C_new = C_old + ΔA·W^T`) with row Merkle verification yields **`4.99×–9.86×`** (`71.51` effective

GMAC/s) single-core on DGX Spark and **7.37×–12.25×** (80.81 effective GMAC/s) on an Apple M2 4× P-core pool, with **100% CD-22 bit-exact parity** and zero unsafe Rust (#![forbid(unsafe_code)]).

- **Game #3 — Simultaneous Full-APU Co-Execution with Zero CUDA / Zero nvidia.ko (gpu/):** Running pure-Omega ollm-gb10 across all **48 SM121 Blackwell SMs** over /dev/aien-gpu (0 /dev/nvidia *nodes,* 0 *nvidia* kernel modules) at the exact same wall-clock instant all **20/20 Grace ARMv9 cores** saturate E8_I8 + pure-OM-2 FIPS 180-4 SHA-256 Merkle verification sustains **467.26 TF** (93.45% of 500 TF peak, **98.42% retention** vs standalone GPU; in-register OMMA.SF.16864 instruction-rate microbenchmark) simultaneously with **132.46 Gmac/s** (**95.58% retention**, **28.7×** 1T Rust) and **2,069.6 MB/s** live SHA-256 Merkle hashing (0 bit mismatches across 449,318 timed matmuls = 3.77T exact MACs).
- **Physical Topology Isolation & Zero-Allocation Frame Pacing (omega-play + /aien/):** Correcting DGX Spark's interleaved big.LITTLE cluster masks (bg_mask=0x07c1f A725 0–4,10–14; game_mask=0xf83e0 X925 5–9,15–19) eliminates **100% of frame stutters** (0 stutters >25 ms vs 9 on stock Linux CFS across 900 frames @ 60 Hz under 24 background threads), raising 0.1% low frame rate from 6.52 FPS to **51.75 FPS** (**7.94×** CFS) and 1% low from 19.82 FPS to **55.41 FPS**. In Chrome on Apple M2, Omega's flat typed-array Zero-Alloc Arena paces 12,000 live entities at **0.3–0.9 ms** CPU work (1.7 ms p99, 0 missed frames) versus 13.5 ms p99 under standard heap allocation.

---

1. Trust Anchor, Hardware Platforms & Verification Methodology

1.1 Reproducible Bootstrap Chain

All pure-Omega binaries in this report are compiled by the seeded Omega compiler (seed/out/omc), grown deterministically from 408 hand-written bytes (seed0 -> hex1 -> asm2 -> om4 -> omc) with zero C on the compiler bootstrap path:

- **DGX Spark pinned omc (linux-aarch64):** 85b84c6de85083e4d9d2368015c1ddcab3d863aada922be0e63d774dddf23134 (2× fixpoint verified)
- **Apple M2 pinned omc (darwin-arm64, target a64macos):** 68a590c37dfb1ec37a183f0032e02db1570db299266f574f65556edfc70feb9c
- **4-Way OP_SELF_TEST Determinism SHA-256 (omega_uor_worker):** 26ce9cfdb6ebf0e57776888456c9fda134451233e572188025aa88d00af74ce7 (bit-identical across linux-aarch64, linux-x86_64 via qemu-x86_64, omega-uor-x925-linux-aarch64, and darwin-arm64)

1.2 Hardware Testbeds & Evidence Classes

- **MEASURED_DGX_SPARK_GRACE_A64 / MEASURED_DGX_SPARK_GB10:** NVIDIA DGX Spark (spark-b87b, Linux 7.0.0-1019-nvidia aarch64), GB10 Blackwell GPU (48× SM121 SMs, 24 MiB L2, 500 TF NVFP4 peak) + Grace ARMv9 CPU (10× Cortex-X925 CPUs 5–9,15–19 @ 3.90 GHz + 10× Cortex-A725 CPUs 0–4,10–14), unified 128 GB LPDDR5X (273 GB/s), booted on sovereign kernel module /dev/aien-gpu (0 /dev/nvidia *nodes,* 0 *nvidia* modules). GPU trials ran inside exclusive spark-window 20261126T232314Z-b68099 (verdict: quiet machine, nvrm_errors_delta=0).
- **MEASURED_MACBOOK_M2_ARM64:** Apple Silicon MacBook Air M2 (Mac14,2, macOS ARM64, 4× P-cores + 4× E-cores, rustc --release -C target-cpu=native, Chrome 158 for live browser workbench benchmarks).
- **Upstream Reference Oracle:** UOR-Foundation/uor-matmul commit 2932286c4ff2718b8e100c4a192df23457b25108, compiled with RUSTFLAGS="-C target-cpu=native" cargo build --release, paired via interleaved A/B trials and checked bit-for-bit (CD-22 parity).

---

2. Pillar I: Closing the Dense CPU Scalar Gap via ARM64 NEON SDOT in seed/ng

Status: VERIFIED on omega-mv feature branch feat/a6-svec-sdot (1da54a5dedec49e4b23c8acac472129304e3c5b9, awaiting PR/operator review before merge to main).

2.1 Architecture

Prior to this work, `seed/ng/src/a64.om` emitted 128-bit floating-point vectors (`vec f32 4` , `vec f64 2`) but fell back to scalar instructions for integer dot products. We extended `a64.om` (`+859/−22` lines) and `ngrun.om` (`+87/−8` lines) with:

1. **17 ARM64 NEON & DotProd Machine Encodings** (forms `181..197` + kind 24 `a6_qpoff` + `.B[0..15]` lane indexing): `sdot4s` (`0x4e809400`), `udot4s` (`0x6e809400`), `add4s` , `sub4s` , `mul4s` , `eor16b` , `addv4s` (`0x4eb1b800`), `dup4s` , `dup16b` , `smovxs` , `umovws` , `smovxb` , `umovwb` , `ld1b16` , `ld1b16p` (`0x4cdf7000`), `ldpq` (`0xad400000`), and `ldpqpo` (`0xacc00000`).
2. **128-bit Integer Vectors & Horizontal Reductions (`a6_svec` + `a6_vred`):** Full support for `vec i8 16` , `vec u8 16` , `vec i32 4` , `vec u32 4` , and `vreduce` via `ADDV Sd, Vn.4S` + `SMOV` / `UMOV` . `a6_hasf` is scoped strictly to floating-point vectors so pure-integer modules emit zero `MSR FPCR` instructions.
3. **Trap-Exact `Q8_dot` NEON `SDOT` Loop Vectorization (`a6_sd_scan` / `a6_sd_emit`):** Recognizes standard pure-Omega `i8 * i8 -> i64/i32` dot-product loops (slice or fixed global array, rotated or unrotated, with or without `chk.fuel` / `chk.bounds` / `chk.mul` / `chk.add`), emits runtime bounds/fuel/overflow pre-guards that fall back cleanly to the exact scalar loop on any failing precondition, and executes a 4×-unrolled 64-byte/iteration `LDP Q` + `SDOT v0..v3.4s` pipeline clamped to `<= 65536` (`i64`) or `<= 32768` (checked `i32`) elements per chunk so 32-bit lane accumulators can never overflow.

2.2 Verification & Paired Hardware Benchmark (`MEASURED_DGX_SPARK_GRACE_A64`)

- **Assembler Referee (`seed/ng/a64/gate.om` vs GNU `objdump`): `PASS`** (`289/289` encodings match, `46/46` refusals, `2/2` mutants killed).
- **Full Compiler Regression & Mutation Suite (`seed/ng/a64/rgate.om`): `PASS`** (`14/14` mkir, `94/94` ngrun incl. `v-i32-ops` , `8/8` refusals, `48/48` front incl. `q8-sdot-parity` , `1000/1000` random seeds, `66/66` float edges, `1000/1000` float seeds, `91/91` mutants killed).
- **Exact Trap Controls (`omc` vs `ng`):** `trap_bounds` `103==103` , `trap_fuel` `104==104` , `trap_ovf` `101==101` , `chunk_70k` (`n=70000`) `0==0` .

Paired interleaved benchmark on Cortex-X925 (`taskset -c 5` , 3.90 GHz, 3 interleaved blocks × 7 rounds = 21 reps/arm):

Vector Length	Exact Sum	`seed/out/omc` Scalar (ns/call)	`seed/ng` NEON `SDOT` (ns/call)	`gcc -O3` Unchecked `vdotq_s32` (ns/call)	Speedup vs Scalar `omc`	Ratio vs `gcc -O3`
`n = 4,096`	`-64,686`	`3,055.132` (min `3,051.095`)	**`35.928`** (min `35.751`)	`33.472` (min `33.388`)	**`85.034×`**	`1.0734×` (includes runtime traps)
`n = 16,384`	`-1,182,067`	`12,126.786` (min `12,107.599`)	**`137.229`** (min `136.924`)	`137.677` (min `136.137`)	**`88.369×`**	**`0.9967×` (beats `gcc -O3` median)**

3. Pillar II: Game #1 — Weight-to-Code Partial-Evaluating `E8_I8` JIT

Status: `VERIFIED` , `MERGED` , & `PUBLISHED` (`omega-observer` @ `d079300` , receipt `uor_unplayable_games_receipt.json`).

3.1 Why Standard Runtimes Cannot Play This Game

Standard inference engines stream quantized weights from RAM/cache into vector registers on every call and run generic multiply-accumulate loops over zero weights alike. Because `UOR-Foundation/uor-matmul` 's `E8_I8` (`Book<256, 8>`) lattice entries consist strictly of `0` , `±16` , and `±32` , our pure-OM-2 partial evaluator (`om/e8_jit_gen.om`) reads the codebook and weight indices once and emits a specialized `.om` program in which:

- Every zero weight (`36.1%` naturally in `E8_I8` , up to `84.7%` with `75%` zero codes) is **erased completely from the binary** — zero instructions emitted, zero branches taken.

- Every nonzero weight is factored into pure two's-complement bit-shifts (`<< 4` for `±16`, `<< 5` for `±32`) and additions/subtractions — **zero weight RAM loads** and **zero integer multiplies** in compute.
- Pinned `seed/out/omc` compiles the emitted source to native machine code in **`6.37–143.07 ms`** (break-even in `144–1,494` calls).

3.2 Measured Results (1T vs 1T vs LLVM Rust `uor-matmul`, 100% `CD-22 PASS`)

Platform	Shape (`M×K×N`)	Zero Codes	Weights Erased	`uor-matmul` Rust (us)	Omega JIT (us)	Speedup	Compile Time (ms)	Binary (KB)	Break-Even Calls
DGX Spark X925	17×128×131	0%	36.2%	61.55	34.45	**1.79×**	23.58	90.5	870
DGX Spark X925	17×128×131	75%	84.7%	61.62	17.28	**3.57×**	6.37	26.5	144
DGX Spark X925	1×256×256	0%	36.1%	71.15	9.71	**7.33×**	91.77	336.4	1,494
DGX Spark X925	1×256×256	75%	84.3%	71.57	6.29	**11.38×**	30.51	89.8	468
DGX Spark X925	16×128×256	0%	36.1%	88.67	57.70	**1.54×**	45.26	172.8	1,462
DGX Spark X925	16×128×256	75%	83.9%	88.86	26.78	**3.32×**	15.58	50.3	252
DGX Spark X925	1×512×256	0%	36.1%	143.28	18.53	**7.73×**	143.07	663.4	1,147
DGX Spark X925	1×512×256	75%	84.1%	144.77	**6.75**	**21.44×**	21.82	172.2	**159**
Apple M2	1×512×256	0%	36.1%	387.92	161.75	**2.40×**	173.43	1,354.7	767
Apple M2	1×512×256	75%	84.1%	318.75	**44.17**	**7.22×**	64.23	371.0	234
Apple M2 (honest reg.)	16×128×256	0%	36.1%	163.21	324.21	0.50×	72.06	371.0	N/A (L1i spill; `1.72×` at 75%)

4. Pillar III: Game #2 — Algebraic 4×-Unrolled `E8_I8`, Exact Merkle Delta-Matmul & Multi-Core Scaling

Status: `VERIFIED`, `MERGED`, & `PUBLISHED` (`omega-observer` @ `d079300`).

4.1 Algebraic Slab + Exact Integer Linearity (`C_new = C_old + ΔA · W^T`)

1. **Single-Core `omega_uor_worker.om` (Game #2A):** Pinning hot buffers in OM-2 callee-saved globals (`X18–X26`), computing the 256-entry E8 activation slab via pure additions/shifts (zero multiplies), transposing `W` to stride-1 block-major layout, and 4×-unrolling the column accumulation flips pure-OM-2 ahead of `--release -C target-cpu=native` LLVM Rust `uor-matmul` even on single-core workloads.
2. **Exact Algebraic Merkle Delta-Matmul (Game #2B):** Because exact `i8 × i8 -> i64` matrix multiplication is strictly linear over the integers (zero floating-point rounding drift!), incremental state updates obey $\mathbf{C_{new} = C_{old} + (A_{new} - A_{old})\ W^T}$ bit-for-bit. Hashing rows with FNV-1a + byte-exact verification skips unchanged rows and skips unchanged 8-element activation blocks inside dirty rows, delivering full `C_new` bit-identical to full recomputation.

4.2 Measured Throughput & Speedup Summary (100% `CD-22 PASS`)

Game	Platform & Concurrency	Shape (`M×K×N`)	`uor-matmul` 1T (us)	Pure-OM-2 (us)	Speedup vs 1T Rust	Effective GMAC/s
2A: 1T Stateless	DGX Spark X925 (1T)	`8×512×2048`	`1,836.64`	**730.55**	**2.51×**	11.48
2A: 1T Cached-W	DGX Spark X925 (1T)	`8×512×2048`	`1,836.64`	**622.39**	**2.95×**	13.48
2A: 1T Cached-W	DGX Spark X925 (1T)	`1×1024×4096`	`351.99`	**282.50**	**1.25×**	**14.85**
2A: 1T Cached-W	DGX Spark X925 (1T)	`16×512×1024`	`1,331.40`	**687.83**	**1.94×**	12.20
2B: 1T Merkle Delta (12.5% rows)	DGX Spark X925 (1T)	`16×512×1024`	`1,331.28`	**135.09**	**9.86×**	62.10
2B: 1T Merkle Delta (9.4% rows)	DGX Spark X925 (1T)	`32×512×2048`	`4,271.11`	**469.24**	**9.10×**	**71.51**
2C: 10× X925 Parallel Cached	DGX Spark 10× X925	`32×512×2048`	`4,305.24`	**1,062.04**	**4.05×**	**31.59**
2B: 1T Merkle Delta (9.4% rows)	Apple M2 (1T)	`32×512×2048`	`5,074.75`	**755.50**	**6.72×**	44.41
2C: 4× M2 Pool Cached	Apple M2 4× P-cores	`8×512×2048`	`2,173.75`	**555.96**	**3.91×**	15.09
2C: 4× M2 Parallel Delta (9.4%)	Apple M2 4× P-cores	`32×512×2048`	`5,085.71`	**415.21**	**12.25×**	**80.81**

5. Pillar IV: Game #3 — Simultaneous Full-APU Co-Execution with Zero CUDA / Zero `nvidia.ko`

Status: `VERIFIED`, `MERGED`, & `PUBLISHED` (`omega-observer` @ `d079300`, receipt `apu_gb10_coexec_receipt.json`).

5.1 Experimental Design & Strict Overlap Proof

On DGX Spark, GB10's 48 SM121 SMs and Grace's 20 ARMv9 cores share a single coherent package. Conventional CUDA stacks spin host threads inside proprietary runtime glue and leave Grace largely idle during GPU runs. Using our sovereign `/dev/aien-gpu` driver (`0` `/dev/nvidia`, `0` *`nvidia`* modules, `ldd: not a dynamic executable` across all binaries), we ran:

- **GPU Half (`ollm-gb10`, `a7b99ea9...`):** SM121 `OMMA.SF.16864.F32.E2M1.E2M1.UE4M3.4X` in-register Tensor Core instruction-rate microbenchmark (slot 84, `500 TF` hardware peak; poisons and checks 64 multi-CTA/warp output lanes + 4 NaN sentinels = `68` checked lanes per run; explicitly an in-register Tensor Core instruction-rate microbenchmark, not a global-memory DRAM GEMM).
- **CPU Half (`20×` pinned `omega-uor-worker` on CPUs `0..19` + pure-OM-2 FIPS 180-4 SHA-256 `om_sha256_merkle`):** All 20 Grace cores (`10× X925` + `10× A725`) continuously compute CD-22 exact `E8_I8 8×512×2048` matmuls, check every 131,072-byte output bit-for-bit against the golden digest (`25954c79...`), and hash every output into a 20-leaf SHA-256 Merkle tree cross-verified by pure-OM-2 `om_sha256_merkle` (`d4a8c931...`, NIST `"abc"` PASS).
- **Strict `CLOCK_MONOTONIC` Overlap Accounting:** CPU workers saturate all cores `~252 ms` **before** `ollm-gb10` starts and continue `~250 ms` **after** `ollm-gb10` exits. Reported simultaneous CPU throughput counts **strictly** ops wholly contained inside the interior active GPU window `[t_gpu_start + 200 ms, t_gpu_end − 100 ms]`.

5.2 Measured Standalone vs Simultaneous Full-APU Results (3-Trial Medians)

Arm	Configuration	GPU NVFP4 Median (TF)	GPU % of 500 TF Peak	GPU Retention	CPU `E8_I8` Median (Gmac/s)	CPU Breakdown (X925 / A725)	CPU Retention	SHA-256 Merkle Rate (MB/s)	Parity Verdict
Arm 0	Canonical GPU probe (`K=8192` , `20` iters)	`453.96`	`90.79%`	—	—	—	—	—	`8192/8192 bad=0 PASS`
Arm 1a	Standalone GPU (`K=8192` , `3000` iters)	`452.05`	`90.41%`	`100.00%`	idle	—	—	—	`3/3 PASS`
Arm 1b	Standalone GPU (`K=131072` , `187` iters)	`474.75`	`94.95%`	`100.00%`	idle	—	—	—	`3/3 PASS`
Arm 2	Standalone CPU (`20/20` Grace cores)	idle	—	—	`138.58`	`97.24` / `41.34`	`100.00%`	`2,165.4`	`0` err / `98,329` ops `PASS`
Arm 3a	Simultaneous (`20/20` cores + fine `K=8192`)	`238.74`	`47.74%`	`52.81%`	`132.37`	`93.38` / `38.99`	`95.52%`	`2,068.3`	`3/3` GPU + `155,233` ops `PASS`
Arm 3b	Simultaneous (`19` workers + A725 CPU 14 feeder, `K=8192`)	`319.21` (pk `400.48`)	`63.84%`	`70.61%`	`129.29`	`93.42` / `35.87`	`93.30%`	`2,020.2`	`3/3` GPU + `113,620` ops `PASS`
Arm 3c	**Simultaneous Full-APU (`20/20` cores + coarse `K=131072`)**	**`467.26`**	**`93.45%`**	**`98.42%`**	**`132.46`**	**`93.15` / `39.02`**	**`95.58%`**	**`2,069.6`**	**`3/3` GPU + `82,136` ops `PASS`**

Whole-System Power Receipt: `spark-receipt` (`~/yardsticks/receipts/2026-11-26-apu-uor-coexec.json`) logged `101.0 W` whole-UPS mean vs `62.0 W` idle (`2,254.09 J` incremental across `57.797 s` , `lock_holders_max=1` , `nvrm_errors_delta=0`). Across all arms, **`449,318 / 449,318`** timed `E8_I8` matmuls (`3.77T` exact MACs) and **`16 / 16`** GB10 SM121 runs passed with zero errors.

---

6. Pillar V: Physical Grace Topology Isolation & Zero-Alloc Real-Time Pacing

1. **Physical DGX Spark Grace Topology (`omega-play` @ `7dba48c5`):** DGX Spark interleaves its efficiency and performance cores across two clusters (`A725` : `0-4,10-14` = `0x07c1f` ; `X925` : `5-9,15-19` = `0xf83e0`). Fixing `om/sched_plan.om` (`SCHED_TOPO_ARM_BIGLITTLE` , `render=5` , `main=6` , `audio=0`) and verifying via `gate.sh` (`15/15 PASS` , `DRY_RUN_READONLY hw_writes=0`) yielded on `bench_sched_isolation` (900 frames @ 60 Hz under 24 competing background threads):
   - **Stutters (`>25 ms`):** `0` vs `9` on stock Linux CFS
   - **`1%` low FPS:** `55.41` vs `19.82 FPS` (`2.80×`)
   - **`0.1%` low FPS:** `51.75` vs `6.52 FPS` (**`7.94×`**)
   - **`p99` frame time:** **`17.00 ms`** vs `25.00 ms`
2. **Live Browser Zero-Allocation Arena & 4K Merkle Studio (`https://www.aienos.com/aien/` , `3fc0b789` , Chrome 158 on Apple M2):**
   - **12,000 live entities:** `0.3–0.9 ms` CPU work per frame, `1.7 ms` `p99` , `0` missed frames vs `13.5 ms` `p99` under standard object-allocating heap churn.

- **60-scene WebCrypto FIPS 180-4 4K Merkle pass:** `0.3 ms` surgical verification (only modified scenes re-rendered; untouched scenes pass by cryptographic identity).

7. Honest Limits, Regressions & Scope Boundaries

In accordance with Omega's strict evidence discipline, every known limit and regression is reported alongside the wins:

1. **SM121 NVFP4 Microbenchmark Scope:** Arm 0 / Arm 1 / Arm 3 GPU throughput (`452–474.75 TF`) measures in-register `OMMA.SF.16864` Tensor Core instruction rate with `68` -lane verification, **not** global-memory LPDDR5X-bound DRAM GEMM.
2. **Host-Feeder Contention at Fine Launch Granularity:** Launching `0.60 ms` GPU kernels (`K=8192` , `1,667` kicks/s) while all 20 Grace cores are 100% busy with `nice 0` workers drops GPU throughput to `238.74 TF` (`52.81%` , Arm 3a) due to Linux CFS preempting the host submission thread. Coarse `K=131072` launches (`9.26 ms` /launch, Arm 3c) eliminate this contention completely (`98.42%` GPU + `95.58%` CPU retention).
3. **Apple M2 JIT L1i Spill:** `16×128×256` at `0%` zero codes (`371 KB` unpruned straight-line binary) runs `0.50×` of LLVM Rust on M2 due to L1i eviction across 16 rows; at `75%` zero codes (`113 KB`) it flips back to `1.72×` faster.
4. **Stateless Pipe & Multi-Core IPC Floor:** Stateless `1×1024×4096` (`M=1`) runs `0.39×` on Spark and `0.17×` on M2 because piping and transposing 512 KB of `W` for a single row dominates compute (caching `W` flips Spark to `1.25×` / `14.85 GMAC/s`). Similarly, `om/omega_uor_x925.om` 's 9-pipe fanout + cross-cluster wakeup adds `~400–600 us` IPC overhead, regressing tiny `M=1` shapes (`0.38×–0.54×`) while winning large shapes (`4.05×` , `31.59 GMAC/s`).
5. `feat/a6-svec-sdot` **Branch Status:** The ARM64 NEON `SDOT` compiler upgrade (`1da54a5de`) is `VERIFIED` on DGX Spark on a feature branch; `omega-mv` `main` remains clean and untouched pending PR/operator review.
6. **Hardware NVENC Video Transcoding:** Remains `BLOCKED_HARDWARE` while Spark boots sovereign `aien_gpu.ko` (zero `nvidia.ko`); pure-Omega CPU heterogeneous work-stealing + 4K Merkle surgical verification operate independently of NVENC.

8. Independent Reproduction Commands

```
# 1. Verify omega-observer receipts & SHA-256 sums
git clone https://interplane.io/drakestapleton/omega-observer.git
cd omega-observer/docs/evidence && shasum -a 256 -c SHA256SUMS.txt

# 2. Rebuild pure-Omega UOR workers & run CD-22 Unplayable Games A/B suite
cd ../.. && OMEGA_OMC=<path-to-pinned-omc> sh om/build.sh
cd rust && RUSTFLAGS="-C target-cpu=native" cargo build --release --bins
taskset -c 5 ./target/release/unplayable_games ../om <path-to-pinned-omc> ../om/out/omega-uor-x925-linux-a
arch64 0

# 3. Verify omega-play physical Grace topology gate (DRY_RUN_READONLY hw_writes=0)
git clone https://github.com/drakestapleton/omega-play.git
cd omega-play && sh gate.sh

# 4. Verify live interactive browser workbench & zero trackers
# Open https://www.aienos.com/aien/ -> DevTools Network -> run /ab, /uor, /merkle
```