RESEARCH · APPLE METAL · ATTENTION AND SMALL OPS

Beyond the Convolution: Attention, Small Ops and Image Models on Apple Silicon

Element-wise ops with full threadgroups · image VAEs on the 3D kernels · Metal FlashAttention inside ggml · FP16 throughout

Exe — Exe AI, developer studio
Measured on Apple M2 Max (38-core GPU, 64 GB unified memory)

343.7 s	1.53×	−43 %	17,688
LTX-2.3 1280×704, 121 frames, whole run; Report 2 ended at 401.6 s	attention call at 14,080 tokens: Metal FlashAttention 320.3 ms against ggml 491.3 ms	Qwen-Image sampling step, 13.80 s → 7.92 s, short rows bundled	ggml op tests on Metal, all passed, every new path switched on (summary line; full log not kept)

SETUP

Start point	ggml fork `9023480a`, the end state of Report 2 (convolution work done: Winograd by default, fast copies)
End point	ggml fork `34e0dc1c` · engine `82a37b3` · application build folder `master-1000-82a37b3`
Question	where the time goes once the convolutions are fast — in image models and in the video transformer
Rules	every change in our ggml fork · stable-diffusion.cpp never edited · numbers internal in FP16 (the M2 Max has no native BF16 arithmetic) · every figure from a log or raw file · one run per figure unless stated

1 Where the time went

Profiler	`GGML_METAL_OP_PROFILE=1` (Report 2, Section 12): GPU time per op label, summed per graph
Models	Qwen-Image 2.1 Turbo and Krea 2 Turbo at 1024×1024 (8 steps) · LTX-2.3 at 1280×704 (121 frames)

GRAPH	GPU TIME	LARGEST ITEMS
Qwen-Image, one sampling step	15.26 s	MUL_MAT q8_0 5.93 s (9.6 TFLOP/s) · **MUL 2.68 s (20 GB/s)** · **REPEAT 2.58 s (5 GB/s)** · FLASH_ATTN 1.72 s · **ADD 1.34 s (19 GB/s)**
Qwen-Image, VAE decode	4.76 s	MUL_MAT f16 2.37 s (7.2 TFLOP/s) · IM2COL 1.44 s (44 GB/s)
Krea 2, one sampling step	17.7 s	MUL_MAT f16 9.37 s (10.7 TFLOP/s) · **MUL 2.31 s** · FLASH_ATTN 2.18 s · **REPEAT 2.10 s** · **ADD 1.20 s**
LTX-2.3 1280×704, one sampling step	73.3 s	MUL_MAT q8_0 34.5 s (10.8 TFLOP/s, 80 %) · **FLASH_ATTN_EXT 29.0 s (304 calls, 5.83 TFLOP/s, 43 %)** · **CONT f32 3.75 s (416 calls, 36 GB/s)** · ADD 1.94 s

- Matrix products run at 72–80 % of Apple's 13.49 TFLOP/s in every diffusion transformer (the Qwen-Image VAE at 53 %) – not the lever.
- Image models: element-wise ops (MUL, REPEAT, ADD) at 5–20 GB/s take about 40 % of a Qwen step; memory bandwidth is about 400 GB/s (Section 4).
- LTX at 1280×704: attention is 40 % of a step at 43 % of peak (Section 6); one family of copies runs at 36 GB/s (Section 5).
- Audio decoder (Report 2): SCALE at 11 GB/s, IM2COL at 28 GB/s (Section 2).
- Logs: `models-…/*_unary-prof_*.log`, `…_fork-e78b5f3-prof_…1280x704…op-profile.log`.

2 Element-wise ops and im2col with full threadgroups

OP (FORK COMMIT)	BEFORE	AFTER
Contiguous unary ops: SCALE, SIN, COS, EXP, GELU, SQR, SQRT, LOG, CLAMP … (`b111ece0`)	one threadgroup per row; the audio decoder reshapes to rows of length 1 → one thread per threadgroup	flat over all elements, 256 threads per threadgroup
IM2COL (`ebcf65a4`)	neighbouring lanes write C·H·W·OH·OW elements apart	one thread per destination element, in destination order · off with `GGML_METAL_IM2COL_FLAT=0`

IM2COL, `TEST-BACKEND-OPS PERF`, TEN 2D SHAPES	BEFORE	AFTER
3×3 kernels	17.2 GB/s	**76–78 GB/s**
5×5 kernels	44.6–51.4 GB/s	72.7–73.2 GB/s

WHOLE RUNS, SAME MODELS, BEFORE → AFTER	BEFORE	AFTER	OUTPUT
LTX-2.3 384×384, audio decode	4.56 s	**2.72 s**	video PSNR 99 dB (identical), audio PCM MD5 identical
LTX-2.3 384×384, generate_video	129.04 s	125.86 s	
Wan 2.2, video decode	23.99 s	23.14 s	pixel-identical
Qwen-Image, VAE decode	11.40 s	**5.66 s**	pixel-identical
Qwen-Image, generate_image	155.55 s	148.33 s	

- Review (high effort): 7 findings; fixed: source contiguity not checked for flat IM2COL, an int overflow of the grid count, a duplicated argument upload, four missing test cases; left: 5 integer divisions per element in the IM2COL kernel (a later speed lever), non-contiguous unary ops with rows of length 1.
- Tests: whole suite 17,652/17,652 after the review; IM2COL 104/104 with the new and the old kernel.

3 Image VAEs on the 3D convolution kernels

- Finding: image VAEs (Qwen-Image, Krea 2) use 2D convolutions; on Metal, stable-diffusion.cpp builds them from IM2COL and MUL_MAT (`src/core/ggml_extend.cpp:409`); ggml's direct `kernel_conv_2d` computes one output per thread, as the 3D kernel did before Report 1.
- Idea: a 2D convolution of one image is a 3D convolution of depth 1 over the same bytes → view CONV_2D as CONV_3D and reuse the native-layout kernels of Report 2.
- Winograd gets the kernel depth as a parameter `KD` (3 or 1): weights, input frames (output frames + KD − 1), GEMM loop and buffer sizes. 3×3 convolutions with C_{in} a multiple of 64 and C_{out} of 128 or 48 run Winograd; the others the direct kernel.
- Code: `ad7be8b7` · a shared fast path for 2D and 3D, its own scratch buffer · off with `GGML_METAL_CONV2D_FAST=0`.
- Reached only with `--conv-direct auto,vae=on`: by default stable-diffusion.cpp keeps IM2COL. Checked: Qwen without the switch on this build, decode 5.66 s and VAE buffer 7,777 MB as before.

TESTS	RESULT
CONV_2D, four settings: default (6 Winograd, 6 direct), all direct, all ccv, off	**2,040/2,040** each, also with a 1 MB scratch
CONV_3D incl. two new depth-1 cases	314/314
All ops	17,666/17,666

IMAGE VAE, 1024²	DECODE BEFORE	DECODE AFTER	VAE BUFFER	OUTPUT AGAINST THE IM2COL PATH
Qwen-Image	5.66 s	**4.24 s**	7,777 → **4,706 MB**	PSNR 65.49 dB, max. difference 6/255
Krea 2	2.67 s	2.42 s	4,417 → 5,096 MB	PSNR 71.73 dB

- Runs: with `--conv-direct auto,vae=on`, one each.
- Krea buffer: +679 MB of Winograd scratch; peak memory 32.12 GB unchanged.
- Review finding: depth 1 also accepts real 1×3×3 nodes of 3D models. Wan 2.2 and LTX-2.3 re-run on this build: both pixel-identical → neither has such a node in the gate.

4 Short rows

- Cause: rotary position embedding in stable-diffusion.cpp reshapes Q and K to rows of length 2 (`model/common/rope.hpp:1111–1125`). ggml's REPEAT, MUL and ADD kernels give each row one threadgroup with as many threads as the row is long → two threads per threadgroup.
- Change (`fac6daa7`): short rows share a threadgroup (second grid dimension), up to 256 threads; long rows unchanged. `850970f8` adds the row-length class (`ne0 <32`, `<256`, `≥256`) to the profile labels.

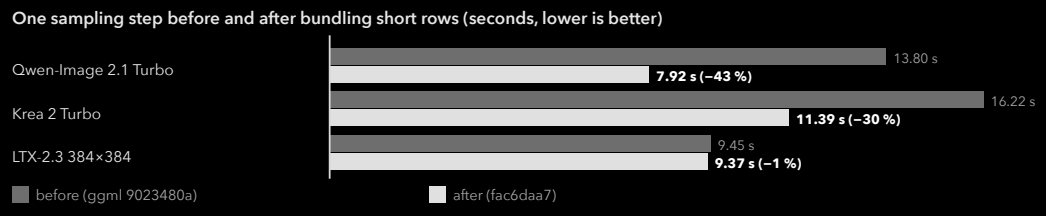


Figure 1. One sampling step, seconds. Qwen and Krea with `--conv-direct auto,vae=on`. Before: ggml `9023480a`; after: build with `fac6daa7`. One run each.

- Whole runs: Qwen generate_image 155.55 → 100.72 s · Krea 195.67 → 157.24 s, image 71.73 dB against the run before.

- LTX-2.3 384×384: 9.45 → 9.37 s per step, video pixel-identical – its transformer has few short rows.
- Tests: all ops 17,666/17,666.

5 Short permuted copies

- Profile at 1280×704: CONT f32, 416 calls, 3.75 s per step at 36 GB/s – although Report 2 made copies fast.
- Diagnosis (`GGML_METAL_GRAPH_DEBUG=2`): the rotary table `pe` [2, 2, d/2, L·heads], permuted (2, 3, 0, 1), 96 times per step. Its innermost source block is four values long; the 32×32 transpose tile of Report 2 ran 1/16 full.
- Change (`d7198edd`): `kernel_cpy_inner` – each thread copies one short contiguous block, with a flat 32-bit index.

LTX-2.3 1280×704	BEFORE	AFTER
That copy op, one call	28.62 ms (30 GB/s)	**2.42 ms (359 GB/s)**
One sampling step	73.75 s	70.24 s
generate_video	402.07 s	**392.45 s**
Output		pixel-identical · CPY tests 208/208, all ops passed

6 Attention

Shape	LTX-2.3 transformer: 32 heads of 128, f32 queries, f16 keys and values, no mask; 14,080 tokens at 1280×704, 2,304 at 384×384
Reference	ggml's f16 flash-attention kernel: 489.80 ms per call at 14,080 tokens, 6.6 TFLOP/s, 49 % of 13.49

6.1 An own kernel: parity, not more

- Design (`metal/fa_long.metal`, laboratory): Q and the output accumulator per simdgroup in registers, a shared K/V tile; checked against a double-precision CPU reference on sampled rows.
- First version: 41.3 ms at 2,304 tokens (laboratory notes; no raw file kept, as for the 15.1 ms below).
- Intermediate findings: element access to the accumulator (`thread_elements`) costs ×6 → rescaling as a matrix multiply with a diagonal matrix (15.1 ms) · a second row block does not fit the registers (×5 slower) · softmax through threadgroup memory instead of on fragments.
- Final: 13.41 ms against 13.33 ms (2,304) and 489.40 ms against 489.80 ms (14,080) – parity at 6.6 TFLOP/s. Without the softmax the kernel still takes 13.5 ms → the limit is loading and occupancy, not the arithmetic.
- Decision: not built in. The missing piece is asynchronous loading, which Metal FlashAttention has.

6.2 Metal FlashAttention

- Source: the generator of Metal FlashAttention as shipped in ccv (`lib/nnc/mfa`, commit `3564f9f`; BSD-3-Clause and MIT).
- Bench (`metal/mfa_bench.mm`, one process): the same shape, error as maximum deviation relative to the output's RMS.

VARIANT	14,080 TOKENS	TFLOP/S	2,304 TOKENS	MAX. ERROR / RMS
ccv's prebuilt library, f16 intermediates	288.43 ms	11.26	7.99 ms	**1.98×10^{-2}**
f32 intermediates, asynchronous loads, f16 Q	331.85 ms	9.79	9.57 ms	1.29×10^{-5}
f32 intermediates, f32 Q read directly (shipped)	**321.73 ms**	**10.10**	9.72 ms	1.29×10^{-5}

- f16 intermediates rejected: 11 % faster, but an error of 2 % of the RMS.
- Build: the current Metal compiler rejects MFA's `__asm` declarations of the asynchronous-copy intrinsics, also offline. Way round: compile the source without them to LLVM IR, rename the declarations to the AIR intrinsics, build the metallib from the IR. Byte-reproducible (`kernels/mfa/tools/build_mfa.sh`); the metallib is embedded.

- Gate (`34e0dc1c`): Apple7 or Apple8 GPU (M1, M2) · f32 Q and output, f16 K and V · head size 128 · no mask, no sinks, no ALiBi, no soft-cap · as many K/V heads as query heads · at least one block of queries · 16-byte alignment, 32-bit index range. Everything else stays on ggml's kernel; `GGML_METAL_FA_MFA=0` switches it off.
- Tests: flash attention 4,997/4,997 · all ops 17,688/17,688.



Figure 2. One attention call at 14,080 tokens, milliseconds, `test-backend-ops perf` (ggml, MFA in ggml) and laboratory benches (own kernel, prebuilt library). At 2,304 tokens: ggml 13.30 ms, MFA in ggml 9.80 ms.

LTX-2.3 1280×704, 121 FRAMES, 4 STEPS	GGML ATTENTION	MFA
One sampling step	70.24 s	**58.48 s**
generate_video	392.45 s	**343.68 s** (−12.4 %)
Video decode	22.15 s	20.93 s
Peak memory	32.11 GB	31.91 GB
Output	PSNR 36.27 dB, SSIM 0.973 — the same scene, not the same pixels	

- Why not identical: another order of floating-point operations in every attention call over four steps; MFA keeps scores, softmax and output in f32 (error against exact 1.3×10^{-5} of the RMS).
- Build of this run: before the final gate checks (GPU family, alignment), otherwise the committed kernel.
- Where it runs (log line `kernel_flash_attn_ext_mfa`): LTX-2.3, Wan 2.2, MiniMax H3, Qwen-Image. Not Krea 2: its 48 query heads share 12 key/value heads, which this kernel does not take.

7 FP16 throughout

- Rule: the M2 Max has no native BF16 arithmetic → every internal number FP16; quantising to Q8_0 is allowed after the conversion to FP16.
- Found (weight statistics in the logs): the VAEs of LTX-2.3, Qwen-Image and Krea 2 and the LTX embedding connectors were BF16; Wan 2.2 F16.
- Conversion (`measurements/bf16_to_f16.py`): new files `*_f16.safetensors`, streamed one tensor at a time, written as `.part` and renamed only without overflow. No overflow anywhere; 203–28,005 values below 6×10^{-8} per file became 0.

EFFECT OF THE F16 FILES	RESULT
Krea 2, decode	2.42 → **1.99 s**
Qwen-Image, LTX-2.3 384	decode times unchanged; images 74–79 dB against BF16
Accuracy, LTX-2.3 384 with exact F32 connectors as reference	F16 **40.14 dB** · BF16 36.40 dB → F16 is closer to the exact result

MINIMAX H3, CONVERTED FROM BF16	TENSORS	SET TO 0	MAX |W|	Q8_0 FILE
Diffusion model	522	6,732	37	35.23 GB
Text encoder (Qwen3-VL 32B)	902	88,821	25	27.49 GB
Audio VAE (from F32)	–	–	–	0.30 GB (F16 GGUF)

- Still BF16: four tensors of the downloaded `qwen-image-2.1-turbo-Q8_0.gguf` (the log compiles `kernel_mul_mm_bf16_f32`) · the LTX 1280×704 runs of Sections 5 and 6 used the BF16 VAE and connector files.

8 Five models before and after

Release	build folder `master-1000-82a37b3` (ggml `34e0dc1c`) in the Exe AI Metal application; the application picks the highest `master-*` folder · only the rows marked "release" ran on it; every other row names its own build
Runs	`sd-cli`, 1024×1024 images with 8 steps, seed 42, `--flash-attn on`; image models with `--conv-direct auto,vae=on`; LTX with `--params-backend te=disk`

MODEL, TASK	BEFORE	BUILD	AFTER	BUILD	OUTPUT CHECK
LTX-2.3, text → video + audio, 1280×704×121	401.64 s	`9023480a`	**343.68 s**	MFA	36.27 dB (Section 6)
LTX-2.3, text → video + audio, 384×384×121	129.04 s	`9023480a`	125.86 s	`b111ece0`	pixel-identical; unchanged within the spread of 125.86–130.05 s
LTX-2.3, image → video + audio, 384×384	163.80 s	`bd04b91`	142.73 s	`e78b5f3`	pixel-identical
Wan 2.2 TI2V 5B, text → video, 832×480×33	223.15 s	`9023480a`	**148.07 s**	release	44.78 dB, SSIM 0.993
Qwen-Image 2.1 Turbo, text → image	155.55 s	`9023480a`	**96.99 s**	release	59.21 dB against the IM2COL path (`1019add`)
Krea 2 Turbo, text → image	195.67 s	`9023480a`	156.25 s	release	71.73 dB; bit-identical between the last two builds
MiniMax H3, first + last frame → video + audio, 864×480, 25 frames requested (39 generated), 20 steps	–		897.26 s	MFA	runs; 35.65 s per step after the first, peak 42.83 GB

- Builds: hashes with eight digits are ggml commits, with seven engine commits (bd04b91 = state of Report 2 before the fast copies, e78b5f3 = ggml `850970f8`); "release" = `master-1000-82a37b3`; "MFA" = the MFA build of Section 6.
- Qwen-Image and Krea 2: the runs before and after differ in switches and VAE files as well; their factors are not attributable to the kernels alone.

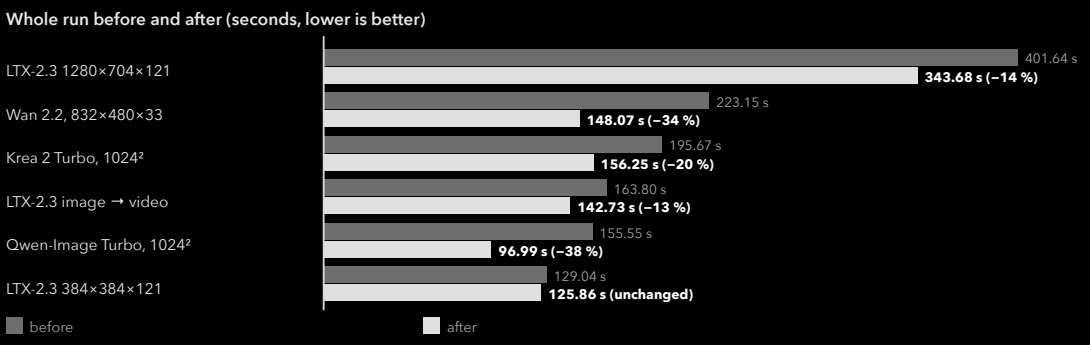


Figure 3. Whole runs (generate_image or generate_video from the log), seconds, one run each; builds as in the table.

- Per step, release against 9023480a: Wan 15.54 → 7.89 s (medians) · Qwen 13.80 → 7.39 s · Krea 16.22 → 11.39 s · LTX 1280×704 73.85 → 58.48 s.
- Wan: the gain lies in the sampling steps – the decode had been fixed already (23.99 s); its split between attention and short rows was not measured.
- MiniMax H3 (full model, 35.2 GB Q8_0): text encoding 67.58 s, first step 113.83 s with loading, sampling 791.29 s, decode 28.29 s; no BF16 weight left; attention on MFA (6,641 tokens). Licence: see the box below.

9 Against the starting point and against Draw Things

LTX-2.3, 384×384, 121 FRAMES, 4 STEPS, SEED 42	WHOLE RUN	PER STEP	VIDEO DECODE	PEAK MEMORY
Draw Things (same model versions)	73.65 s	11.50 s	not reported	27.2 GB

LTX-2.3, 384×384, 121 FRAMES, 4 STEPS, SEED 42	WHOLE RUN	PER STEP	VIDEO DECODE	PEAK MEMORY
stable-diffusion.cpp master-1000, unmodified	512.91 s	10.64 s	377.07 s	44.6 GB
This report	**125.86 s**	**9.45 s**	**4.58 s**	29.68 GB

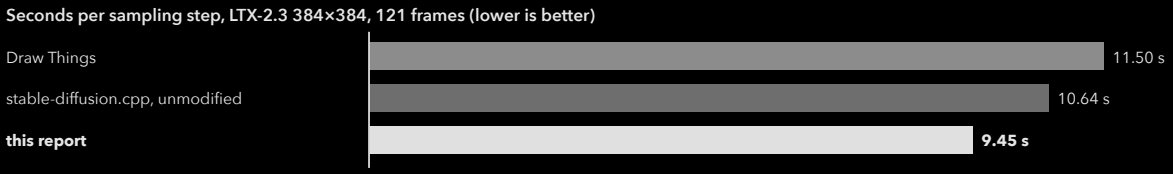


Figure 4. Seconds per sampling step, LTX-2.3 384×384. Draw Things from its own log, mean of its four steps (median 11.22 s); the others from stable-diffusion.cpp's log, the run of the table.

- Per step this build is 18 % faster than Draw Things (16 % against its median); Draw Things ran another quantisation and produced another picture; whole run 52 s slower: about 80 s of our run is reading the weights from the external USB SSD (Report 2, Section 12). Whether Draw Things' files were cached is unknown.
- LTX-2.3 1280×704: unmodified kernels took 3,580.17 s (decode 2,721.06 s; a low-priority run, too slow – at normal priority the decode alone took more than 1,360 s) → 343.68 s with this report.
- Image models, first survey (stable-diffusion.cpp master-975, wall clock with loading, other build): Qwen-Image 2.1 Turbo 1024² 156.2 s (no log kept) → 96.99 s (generate_image, not the same clock).

10 Open points and next steps

OPEN POINT	NEXT STEP	SECTION
Server path	no model run through `sd-server` with the new kernels yet	8
Untested tasks	Qwen-Image editing (reference image + vision projector) · Qwen-Image without Turbo · LTX first + last frame · Wan image → video	8
Application defaults	the application does not yet pass `--conv-direct auto,vae=on`; F16 files and `te=disk` as defaults undecided	3, 7
Remaining BF16	four tensors in the Qwen-Image Turbo file; LTX 1280×704 re-run with the F16 VAE and connectors	7
Repeats	every engine figure here is one run	all
Krea attention	attention with shared key/value heads stays on ggml's kernel (39–40 % of peak)	6
Remaining slow ops	IM2COL divisions per element · non-contiguous unary ops with rows of length 1	2
Memory reserve of the application	the need of MiniMax H3 above its weights is not measured (peak 42.83 GB); the reserve is 2.5 GB	8

MiniMax H3 licence

- The MiniMax H3 licence, in the version consulted for this report, excludes the European Union, the United Kingdom, the Republic of Korea and the United States, and forbids using or displaying the model's outputs outside its territory; a licence can be requested from MiniMax.
- This report therefore gives timings only and shows no output of the model.

Licences and acknowledgements

COMPONENT	ORIGIN
stable-diffusion.cpp, ggml, ccv's 3D convolution kernel	Report 1

COMPONENT	ORIGIN
Metal FlashAttention kernel (`kernels/mfa`), generated by the MFA generator in ccv	ccv (BSD-3-Clause, © 2010 Liu Liu) · Metal FlashAttention (MIT, © 2024 Philip Turner); both texts in `kernels/mfa/LICENSE` and beside the application's binaries
Flat unary ops, destination-ordered IM2COL, CONV_2D on the 3D kernels, Winograd depth parameter, short-row bundling, short permuted copies, the MFA integration and IR build, the own attention kernel, conversions, measurements, analysis	Exe AI, own work
Draw Things	measured comparison of the released application's time and memory only; no code used
GPL-licensed code	none used

References

1. Exe AI, *Fast 3D Convolutions for Local Video Generation on Apple Silicon*, Technical Report 1.
2. Exe AI, *Memory and Speed Together: Optimising Local Video Generation on Apple Silicon*, Technical Report 2.
3. P. Turner, Metal FlashAttention; L. Liu, ccv (`lib/nnc/mfa`), github.com/liuliu/ccv.
4. Apple, M2 Max compute peak of 13.49 TFLOPS (38-core GPU), as published by Apple.
5. Project measurements: `measurements/RESULTS.md`, `measurements/models-…/`, `measurements/tests-…-*/`, `measurements/…_fa_m-fa_test-backend-ops.txt`, step log `paper/paper3-schritte.md`; laboratory raw files `metal/messungen/…_fa_long.txt`, `…_mfa_bench.txt`.

EXE AI · DEVELOPER STUDIO