TINY VLM MODELS ON SNAPDRAGON CHIP USING LLAMA.CPP

CVPR 2026 Tutorial

The IEEE/CVF Conference on Computer Vision and Pattern Recognition 2026

Denver, CO, USA

BUILDING LLAMA.CPP WITH SNAPDRAGON

Overview

llama.cpp provides an easy way to build Snapdragon-based Android device tools using containers.

```
$ git clone https://github.com/ggml-org/llama.cpp.git
$ git submodule update --init --recursive
$ cd llama.cpp
$ docker run -it -u $(id -u):$(id -g) --volume $(pwd):/
workspace --platform linux/amd64 ghcr.io/snapdragon-toolchain/
arm64-android:v0.3
```

Once inside the container, follow the instructions from the llama.cpp repository to build for Snapdragon.

EVALUATION SETUP

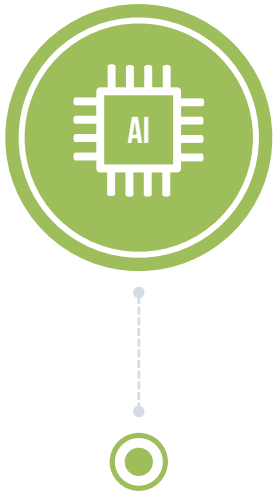
Resources

Compute Device

Snapdragon 8 Elite Mobile (SM-8750)

8x Orion CPU (2 Prime + 6 Performance)

Adreno 830 GPU, Hexagon NPU



Compute Backends

CPU: 2, 4, 6 threads

HTP (Hexagon NPU)

GPU (OpenCL)

Methodology

Model: *Qwen3VL-2B VLM*

Quantization: *Q8_0, Q4_0, IQ4_NL*

Vision Model (mmproj): *FP16*

Prompt

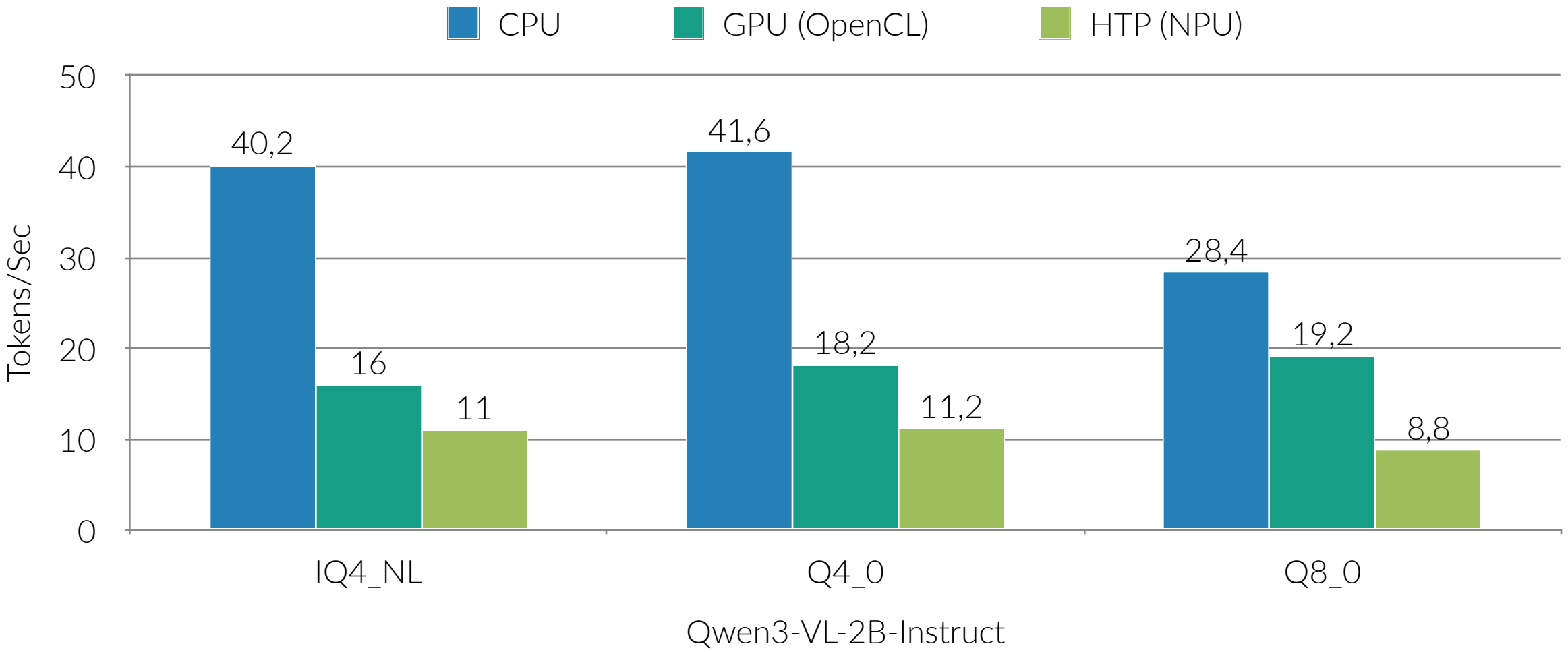
"Describe the image in detail"

Image: 640 x 640

Output capped at 256 tokens

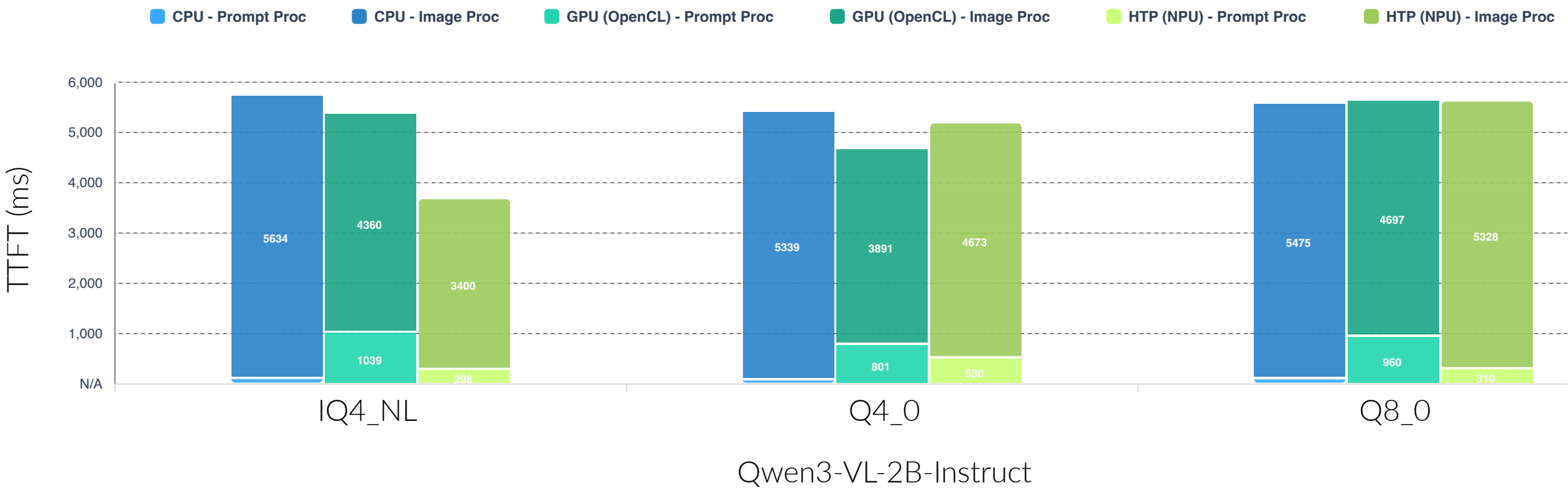
BACKENDS COMPARISON

Output Throughput (tokens/sec) — CPU vs HTP vs GPU



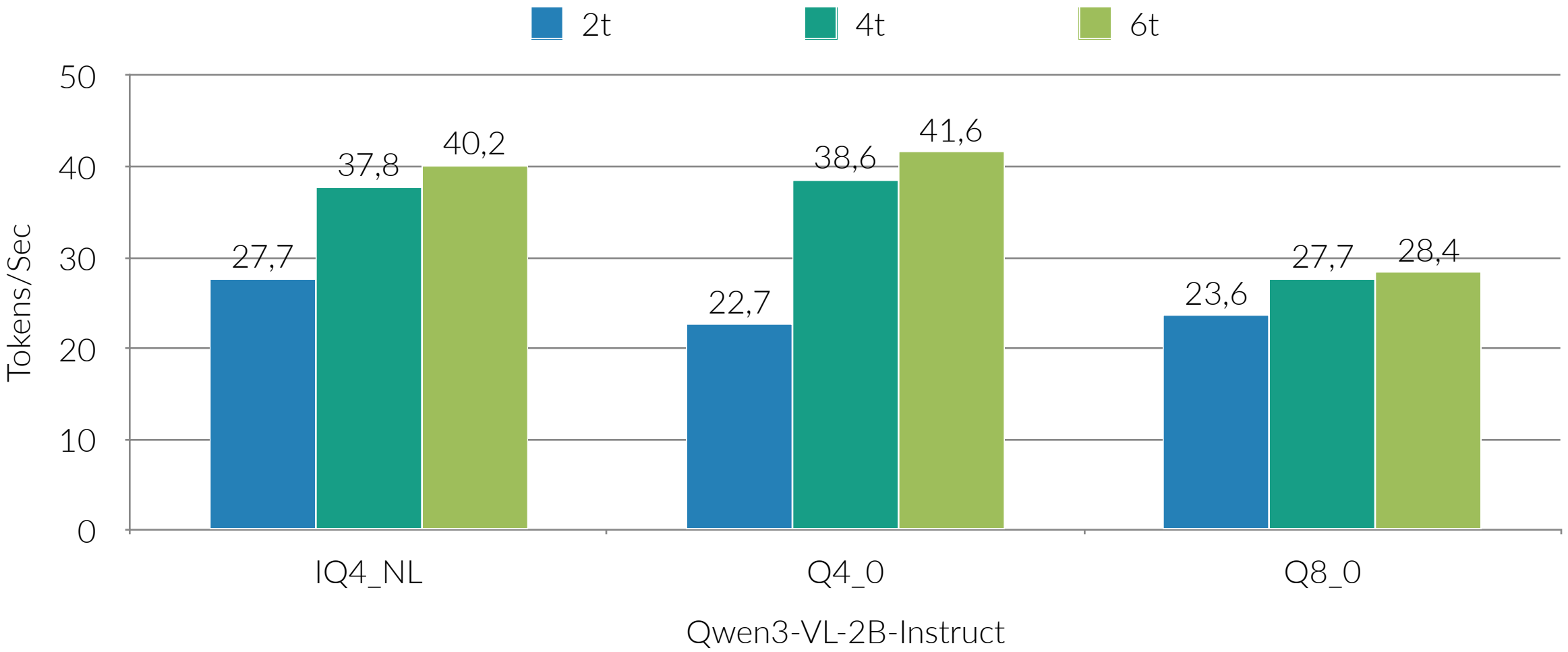
BACKENDS COMPARISON

Time-To-First-Token (sec) — CPU vs HTP vs GPU



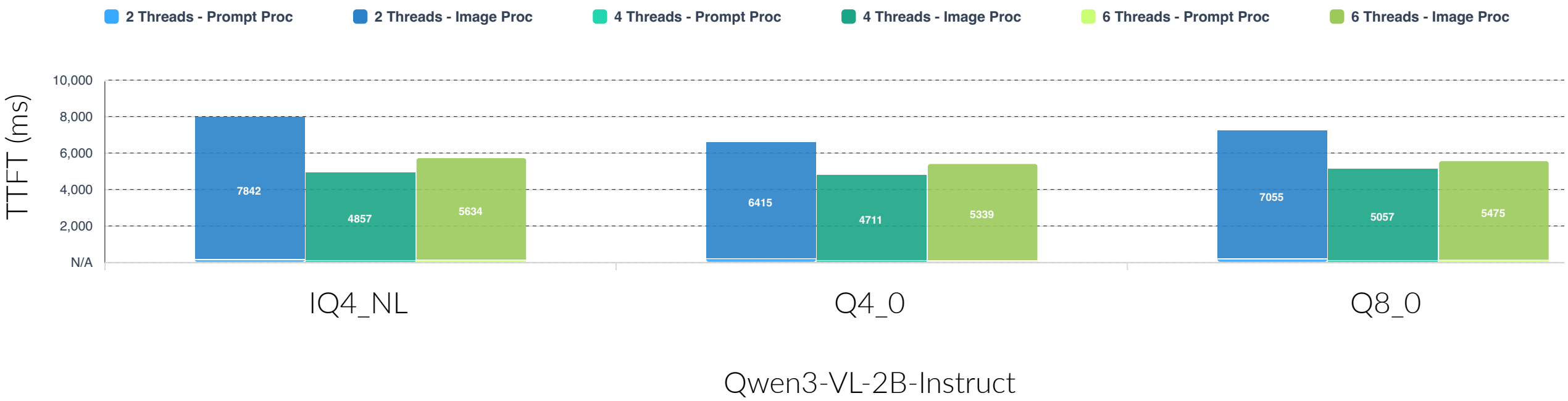
BACKENDS COMPARISON

Output Throughput (tokens/sec) — 2, 4, 6 Threads



BACKENDS COMPARISON

Time-to-First-Token (sec) — 2, 4, 6 Threads



BACKEND COMPARISON ANALYSIS

Key Performance Insights from On-Device VLM Inference

CPU OUTPERFORMS HTP & GPU

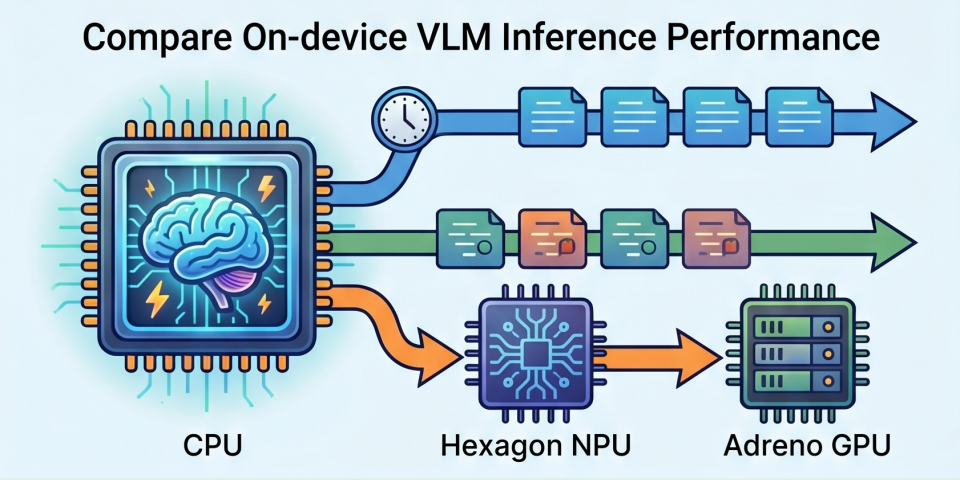
CPU delivers higher throughput and lower latency than both Hexagon NPU (HTP) and Adreno GPU (OpenCL) for this VLM workload.

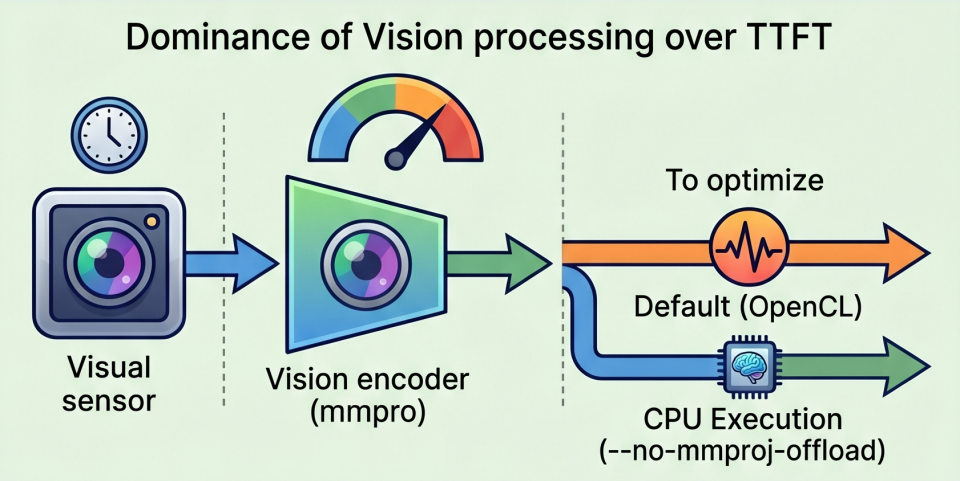
VISION PROCESSING DOMINATES TTFT

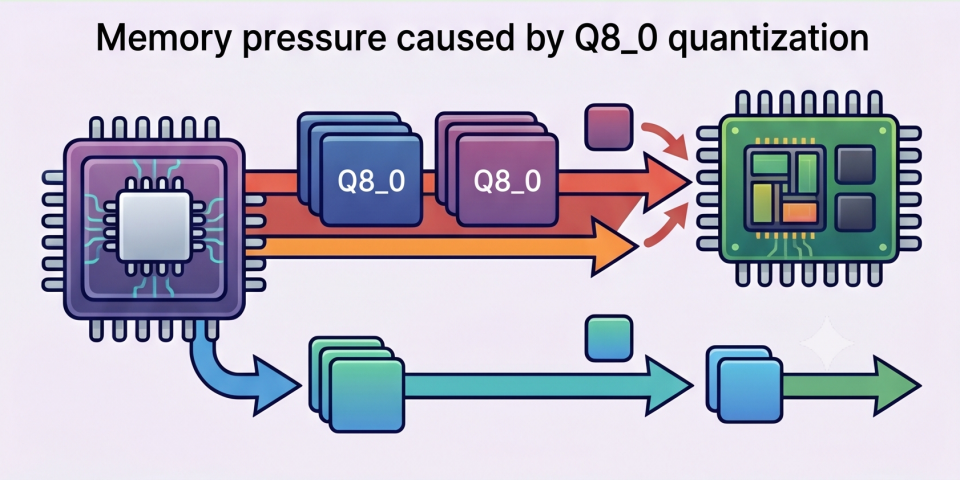
Time-to-first-token is dominated by vision encoding time. The mmproj runs on OpenCL by default, causing high variance. Use **--no-mmproj-offload** to force CPU execution.

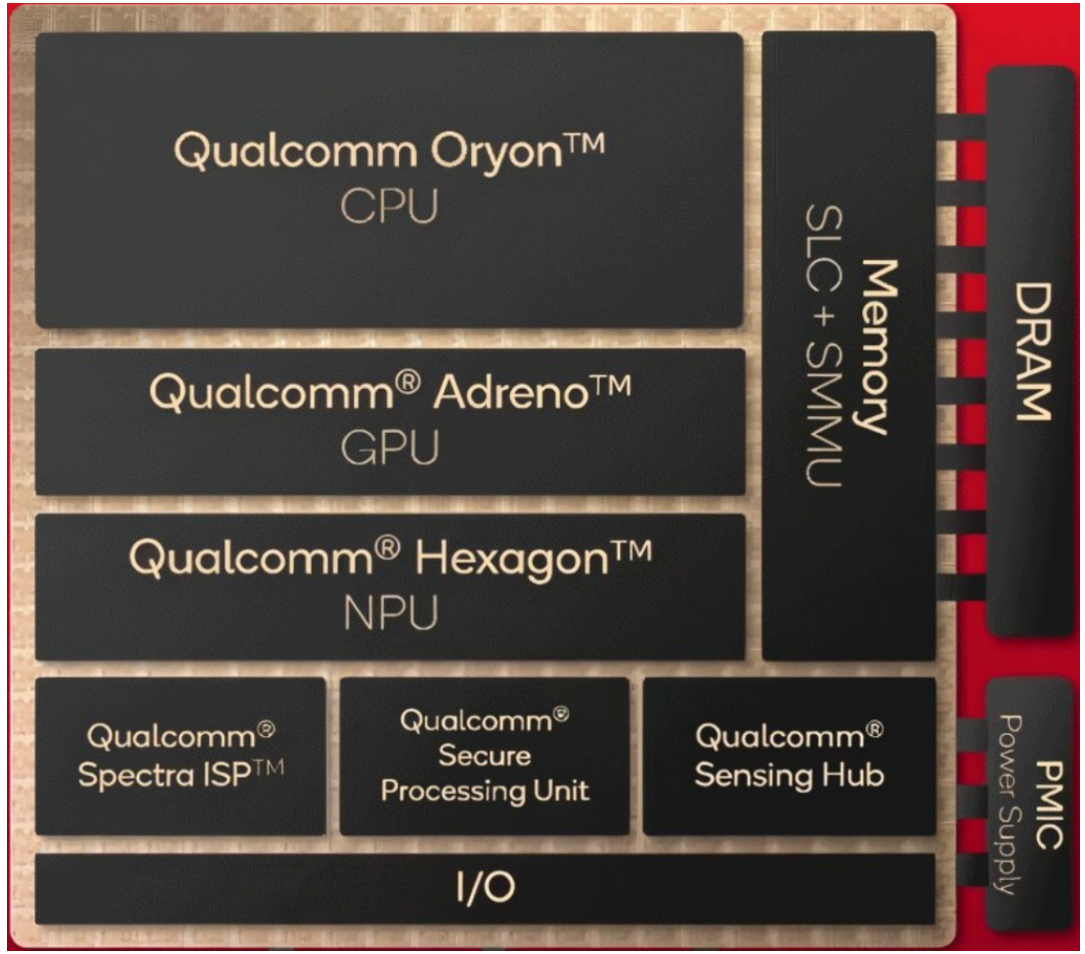
Q8_0 MEMORY PRESSURE

Slowdown in Q8_0 throughput is due to higher memory and bandwidth usage, putting pressure on the unified memory architecture.









MULTIPLE CV TASKS

On a Single SoC

On the same Snapdragon 8 Elite, we plan to run multiple computer vision tasks concurrently:

Object Detection

Identifies and localizes objects in video frames.

Direction of Arrival (DoA)

Estimates the direction of incoming sound sources.

Frame Stitching

Combines multiple frames into a unified view.

Echo Cancellation

Removes acoustic echo from audio streams.

Most tasks are compute-intensive and run on HTP and GPU.

Utilizing CPU for VLM reduces overall system latency.

CPU MASKING STRATEGY

Flags

Use --cpu-mask to select specific cores and --no-mmproj-offload to run vision backbone on CPU.

CPU CORES ORDER

L0 (Core 0) Prime 1	L1 (Core 1) Prime 2	L2 (Core 2) Perf 1	L3 (Core 3) Perf 2
L4 (Core 4) Perf 3	L5 (Core 5) Perf 4	L6 (Core 6) Perf 5	L7 (Core 7) Perf 6

2 PRIME CORE + 2 PERF. CORES (0X0F)

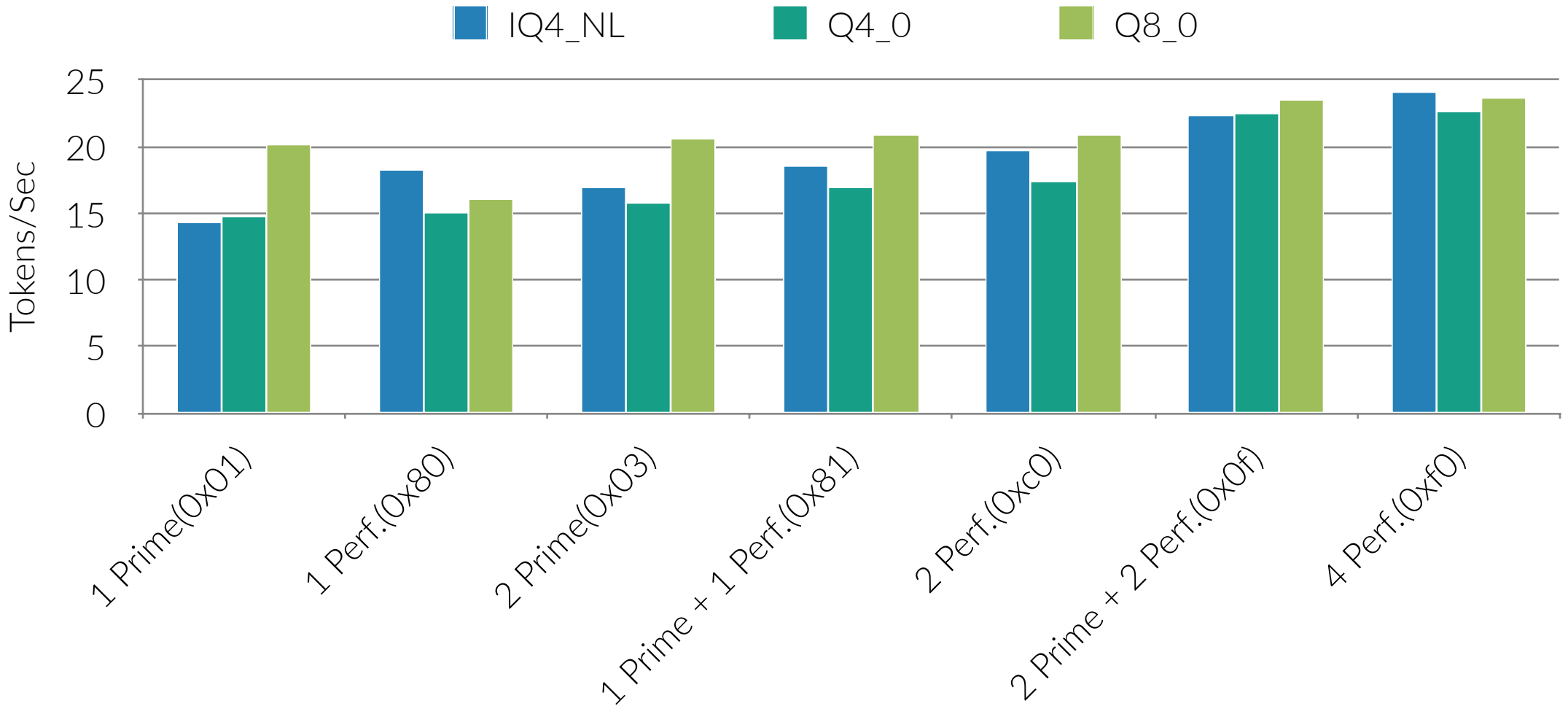
L0 (Core 0) **Prime 1**	L1 (Core 1) **Prime 2**	L2 (Core 2) **Perf 1**	L3 (Core 3) **Perf 2**
L4 (Core 4) Perf 3	L5 (Core 5) Perf 4	L6 (Core 6) Perf 5	L7 (Core 7) Perf 6

4 PERFORMANCE CORES (0XF0)

L0 (Core 0) Prime 1	L1 (Core 1) Prime 2	L2 (Core 2) Perf 1	L3 (Core 3) Perf 2
L4 (Core 4) **Perf 3**	L5 (Core 5) **Perf 4**	L6 (Core 6) **Perf 5**	L7 (Core 7) **Perf 6**

BACKENDS COMPARISON

Output Throughput (tokens/sec) — 2, 4, 6 threads



Qwen3-VL-2B-Instruct

BACKENDS COMPARISON

Time-to-First-Token (sec) — Core Selection Impact

Qwen3-VL-2B-Instruct

FINAL ANALYSIS

Conclusions

1 4 THREADS = SWEET SPOT

Using 4 threads with 2 Prime + 2 Performance cores delivers efficient performance with good latency and power balance.

2 QUANTIZATION PARITY

All quantization types (Q8_0, Q4_0, IQ4_NL) are within 10% of each other. The choice comes down to accuracy vs. model size trade-offs.

3 VISION BACKBONE QUANTIZATION MATTERS

Quantizing mmproj from FP16 to Q8_0 may affect performance since the vision backbone runs entirely on CPU.