

Eseguire Ternary Bonsai 2 27B con llama.cpp

Tag: AI · Guide · llama.cpp · Modelli-locali

Ultimo aggiornamento: 2026-09-20

Eseguire Ternary Bonsai 2 27B con llama.cpp

Guida operativa verificata sul server pcsebus (RTX 5060 Ti 16 GB, Debian 12), aggiornata al 20 settembre 2026 dopo installazione, test funzionali e confronto diretto con Qwen3.8-27B. Fonti: model card HF prism-ml/Ternary-Bonsai-2-27B-gguf, repo PrismML-Eng/Bonsai-demo (fonte ufficiale per il serving), whitepaper bonsai-2-27b-whitepaper.pdf (in /localIA/llama/Bonsai-demo/).

1. Che cos'è Bonsai 2 27B

Modello di Prism ML derivato da Qwen3.8-27B: stessa architettura, pesi ristrutturati in formato ternario $\{-1, 0, +1\}$ con scale FP16 group-wise.

Caratteristica	Valore
Base model	Qwen3.8-27B (27,36B param: backbone 24,35B + embed/LM head 2,54B + vision 0,46B)
Quantizzazione	ternaria, 1,72-1,75 bit/peso reali (senza trucchi nascosti)
Pesi	7,21 GB (PQ2_0) o 5,95 GB (PTQ1_0), contro ~54 GB FP16 (~9x)
Ritenzione qualità	98,2% del FP16 (84,78 media su 14 benchmark thinking)
Contesto	262K token (attenzione ibrida ~75% lineare → KV economica)
Modalità	testo + immagini (vision), thinking mode, tool calling nativo
Licenza	Apache 2.0
Throughput	~47 tok/s su Apple M5 Max; ~44-48 tok/s misurati su RTX 5060 Ti

⚠ Il llama.cpp stock NON può caricarlo: servono i binari del fork PrismML con i kernel ternari (PTQ1_0/PQ2_0). Un errore tipico del runtime sbagliato è unknown quantization format.

2. Installazione su pcsebus

Tutto è già installato in /localIA/llama/Bonsai-demo/ (repo clonata + setup eseguito). Per reinstallare da zero:

```
cd /localIA/llama
git clone https://github.com/PrismML-Eng/Bonsai-demo.git
cd Bonsai-demo
```

```
./setup.sh          # scarica binari llama.cpp fork (bin/cuda) + pesi GGUF + mmproj
```

File scaricati in `models/bonsai2-gguf/27B/`:

File	Contenuto	Dimensione
Ternary-Bonsai-2-27B-PQ2_0.gguf	pesi LM (formato default della demo)	7,21 GB
Ternary-Bonsai-2-27B-PTQ1_0.gguf	pesi LM, alternativa più compatta (decode più veloce, prompt più lento)	5,95 GB
Ternary-Bonsai-2-27B-mmproj-Q8_0.gguf	proiettore vision (serve solo per immagini)	0,63 GB

L'installazione sfrutta i CUDA runtime già presenti sul sistema: il launcher ufficiale usa `LD_LIBRARY_PATH` con `/usr/local/lib/ollama/cuda_v12 (libcudart.so.12)` — senza, il binario non parte.

3. Configurazione su pcsebus

Specchio di quella del server Qwen (`/etc/llama-server.conf` + `/usr/local/bin/llama-start` + `llama-server.service`), con le stesse convenzioni:

Componente	Percorso
Config	<code>/etc/llama-bonsai-server.conf</code>
Script di avvio	<code>/usr/local/bin/llama-bonsai-start</code>
Servizio systemd	<code>llama-bonsai-server.service</code> (enabled)
Binario	<code>/localIA/llama/Bonsai-demo/bin/cuda/llama-server</code> (fork PrismML)

Gestione (identica a llama-server):

```
sudo systemctl start llama-bonsai-server    # avvio
sudo systemctl stop llama-bonsai-server     # stop
journalctl -u llama-bonsai-server -f        # log
```

⚠ Il server Qwen e il server Bonsai sono mutuamente esclusivi: stessa porta (11435) e VRAM insufficiente per entrambi. Per cambiare modello:

```
sudo systemctl stop llama-bonsai-server && sudo systemctl start llama-server    # → qwen
sudo systemctl stop llama-server && sudo systemctl start llama-bonsai-server    # → bonsai
```

Parametri attuali (2026-09-20): `alias bonsai-2-27b`, porta 11435, `ctx 131072`, `KV q4_0`, `-np 2`, `--kv-unified`, `-ngl 999`, `--flash-attn on`, `threads 7`, `mmproj in RAM` (`--no-mmproj-offload`), `thinking con budget 2048` (`--reasoning-budget 2048`), `sampling --temp 1.0 --top-p 0.95 --top-k 20` (identico al qwen).

4. Personalizzazione (cosa toccare nella conf)

Tutto in `/etc/llama-bonsai-server.conf`; dopo ogni modifica `sudo systemctl restart llama-bonsai-server`.

Voglio...	Modifica
Contesto più corto/lungo	CTX= (32768 base; ≥65K solo con KV4 attivo)
KV 4-bit per contesti lunghi	EXTRA_ARGS=(... -ctk q4_0 -ctv q4_0 ...); migliore qualità con make_kv_bias.sh
Peser PTQ1_0 (più piccoli, decode più veloce)	decommentare MODEL=...PTQ1_0.gguf (richiede il download)
Projector vision su GPU (immagini più rapide)	commentare --no-mmproj-offload (+0,63 GB VRAM)
Più/meno thinking	--reasoning-budget N (0 = niente thinking, -1 = illimitato); il budget è un tetto, non un target
Sampling diverso	--temp / --top-p / --top-k / --min-p in EXTRA_ARGS
Porta/alias	PORT= / ALIAS=
Reasoning effort	l'API accetta reasoning_effort: low/medium/high/xhigh (default xhigh; low non supportato, si comporta come xhigh)

Lo script di avvio gestisce LD_LIBRARY_PATH (binari fork + CUDA runtime) e i flag di base; le scelte personali stanno nella conf.

5. Confronto diretto con Qwen3.8-27B (GSQ-RCO, IQ3_S+MTP)

Configurazione identica sui due server (verificata flag per flag): ctx 131K, KV4, np 2 kv-unified, batch, graphs, thinking budget 2048, sampling temp 1.0 / top-p 0.95 / top-k 20. Unica differenza: il qwen usa il draft MTP, che Bonsai non ha.

Test chat (matematica, logica mazza+palla, ragionamento): entrambi corretti sempre. Test coding con suite verificata automaticamente (conversione romani, bugfix ricerca binaria con loop infinito, parsing ordini → top prodotti): 3/3 entrambi, con codice di qualità sovrapponibile (Bonsai usa Decimal per il denaro, Qwen il tie-break deterministico — la soluzione ideale sarebbe la somma delle due).

Metrica	Qwen3.8 GSQ-RCO	Bonsai 2 27B
Decode	~40 tok/s	~44 tok/s
Qualità	riferimento	equivalente nei test (98,2% dichiarato del FP16)
Robustezza thinking	thinking a volte in loop fino a esaurire i token	sempre contenuto prodotto
VRAM a 131K+KV4	15,4 GiB su 16 (al limite)	~10,7 GiB (margine ~5 GiB)
Draft speed	MTP (+ speculativo)	nessuno (esiste il dspark opzionale)

Verdetto pratico: qualità equivalente, Bonsai più veloce, più stabile e con la metà dell'occupazione — su una 16 GB è la scelta migliore.

6. Occupazione di memoria

Formati dei pesi (dal whitepaper):

Formato	bit/peso	Dimensione
FP16 (riferimento)	16,0	~54 GB
Ternario ideale	1,72	5,8 GB
GGUF PTQ1_0	1,75	5,95 GB
GGUF PQ2_0	2,13	7,21 GB

Sul pcsebus (16 GB VRAM), misurato con nvidia-smi:

Scenario	VRAM usata
Bonsai PQ2_0, ctx 32K, mmproj su GPU	10,1 GiB
Bonsai PQ2_0, ctx 131K + KV4, mmproj su GPU	10,7 GiB
Bonsai PQ2_0, ctx 131K + KV4, mmproj in RAM (attuale)	~9,9 GiB
Qwen GSQ-RCO IQ3_S, ctx 131K + KV4	15,4 GiB

KV cache a FP16: ~64 KiB/token (contenuta dall'attention ibrida); con KV4 ~18 KiB/token (~1,8 GiB a 100K). Senza KV4, 131K non entrerebbe nei 16 GB.

7. Note operative

- I due servizi si escludono a vicenda (porta + VRAM): mai avviati insieme.
- Per fermare il server demo standalone della repo (se lanciato a mano): attenzione al `pkill -f` via SSH — il pattern finisce nella cmdline della shell remota e la uccide; usare `pkill -f "cuda/llama[-]server"` (con le parentesi quadre).
- Vision: immagini via `--mmproj`; il projector in RAM rende il prefill immagini più lento (CPU encode) ma il testo non ne risente.
- I test harness usati per il confronto sono in `/tmp/test_llm.py`, `/tmp/test_llm2.py` e `/tmp/test_coding.py` su pcsebus.