Intel Arc B580 for Local LLMs: 12GB VRAM at $250, With Caveats

March 5, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** The Intel Arc B580 gives you 12GB GDDR6 for ~$250, making it the cheapest way to get 12GB VRAM for local LLMs. It runs 7-9B models at 13-15 tok/s via llama.cpp Vulkan, roughly matching an RTX 3060 12GB. The catch: Intel's software stack requires more setup than NVIDIA, Ollama support is indirect through IPEX-LLM, and you'll hit walls that CUDA users never see. Worth it if you're comfortable troubleshooting. Not worth it if you want plug-and-play.

The Intel Arc B580 is the cheapest way to get 12GB of VRAM right now. At ~$250 street price, it undercuts the RTX 3060 12GB by $50-100 on the used market and gives you enough memory to run every 7-9B model without compromise.

The problem isn't the hardware. The hardware is fine. The problem is that NVIDIA has had a decade to build CUDA into the default path for everything, and Intel is still catching up. Running LLMs on an Arc card means picking your way through software stacks that change every few months, dealing with setup steps that CUDA users never think about, and occasionally hitting bugs that make you question your life choices.

I still think it's worth considering, if you go in with the right expectations.

What the B580 Brings to LLM Inference

Spec	Value	Why It Matters
VRAM	12GB GDDR6	Fits 7-9B models at Q4-Q8, some 14B at Q4
Memory bandwidth	456 GB/s	Higher than RTX 3060 (360 GB/s)
XMX engines	160	Intel's matrix math units, analogous to CUDA Tensor Cores
TDP	150W	Reasonable, single 8-pin power
Street price	~$249	Cheapest 12GB card available new

The bandwidth number is the one that matters most for LLM inference. Token generation speed is bottlenecked by how fast the GPU can read model weights from VRAM. The B580's 456 GB/s is 27% more than the RTX 3060's 360 GB/s. In theory, that should translate to faster generation. In practice, software overhead eats some of that advantage.

The Three Software Paths

There are three ways to run LLMs on an Arc B580. They are not equally good.

Path 1: llama.cpp with Vulkan (Easiest, Recommended)

The Vulkan backend in llama.cpp is the simplest path. Vulkan is cross-platform, doesn't require Intel's oneAPI toolkit, and works with standard GPU drivers.

Download a llama.cpp release with Vulkan support, point it at a GGUF model, and go:

```
# Download latest llama.cpp release with Vulkan
./llama-server -m qwen3.5-9b-q4_k_m.gguf -ngl 99 --gpu-backend vulkan
```

Community reports and Phoronix benchmarks consistently show Vulkan outperforming SYCL on the B580. This is ironic given that SYCL is Intel's own stack, but it's the reality as of early 2026.

Path 2: llama.cpp with SYCL (Intel's Stack)

The SYCL backend uses Intel's oneAPI toolkit and gives access to the XMX engines for matrix operations. Setup is heavier:

```
# Install oneAPI Base Toolkit (2025.1+)
# Then build llama.cpp with SYCL
source /opt/intel/oneapi/setvars.sh
cmake -B build -DGGML_SYCL=ON -DCMAKE_C_COMPILER=icx -DCMAKE_CXX_COMPILER=icpx
cmake --build build --config Release -j

# Run with environment variables
ZES_ENABLE_SYSMAN=1 ONEAPI_DEVICE_SELECTOR=level_zero:0 \
  ./build/bin/llama-server -m model.gguf -ngl 99
```

SYCL should be faster in theory because it can use the XMX engines directly. In practice, community testing shows Vulkan consistently beating SYCL on the B580 for llama.cpp inference. Intel is actively working on this, and the gap may close with future oneAPI releases. For now, start with Vulkan.

Path 3: Ollama via IPEX-LLM (Most Setup)

Ollama doesn't natively support Intel Arc GPUs. You can run it through Intel's IPEX-LLM bridge, which wraps Ollama with Intel-optimized backends.

The setup involves:

1. Install oneAPI toolkit
2. Create a conda environment
3. Install IPEX-LLM Ollama package
4. Set environment variables (`OLLAMA_NUM_GPU=999`, `ZES_ENABLE_SYSMAN=1`)
5. Run `init-ollama` to configure
6. Start Ollama with IPEX-LLM backend

It works. Users have reported running Qwen3-8B at ~33 tok/s through this path with 16K context. But there are known issues: model loading failures on some driver versions, bus errors on newer Linux kernels, and the setup is fragile enough that a driver update can break things.

If you need the Ollama interface specifically, this path exists. If you just need to run models, use llama.cpp Vulkan and skip the complexity.

Real Benchmarks

These numbers come from Phoronix/OpenBenchmarking testing llama.cpp with the Vulkan backend on Linux. Text generation at 128 tokens output:

Model	B580 tok/s	Notes
Llama 3.1 8B (Tulu-3)	12.9	Standard 8B performance
Mistral 7B	14.4	Slightly faster due to architecture
Qwen3 8B	13.3	Consistent with other 8B models
DeepSeek-R1-Distill 8B	13.7	Reasoning model, same speed tier

Model	B580 tok/s	Notes
Granite 3B	58.2	Small models fly
GPT-OSS 20B	26.6	MoE architecture, not dense 20B

Prompt processing (prefill) is much faster: 590-640 tok/s for 8B models, over 2,300 tok/s for 3B. The B580's compute is solid for prefill. It's the generation phase where bandwidth becomes the bottleneck.

For the IPEX-LLM/OpenVINO path, Intel's own benchmarks claim significantly higher numbers (150+ tok/s for 7B models). Those numbers appear to come from batched or optimized pipeline scenarios that don't reflect typical single-user interactive chat. The Vulkan numbers above are what you'll see in normal use.

B580 vs RTX 3060 12GB vs RX 7700 XT

	Arc B580	RTX 3060 12GB	RX 7700 XT
VRAM	12GB GDDR6	12GB GDDR6	12GB GDDR6
Bandwidth	456 GB/s	360 GB/s	432 GB/s
Price (new/used)	~$249 new	~$170-220 used	~$350 new
7-9B tok/s	13-15	12-20 (CUDA)	15-22 (ROCm)
Software stack	Vulkan/SYCL	CUDA	ROCm
Ollama support	Via IPEX-LLM bridge	Native	Native (recent)
Setup difficulty	Medium-High	Easy	Medium
Driver maturity	Young	Mature	Improving

The RTX 3060 12GB wins on software maturity. CUDA just works. Every guide and troubleshooting post assumes NVIDIA. If you're buying used and want the smoothest experience, the 3060 is the safer pick at $170-220.

The B580 wins on bandwidth per dollar. 456 GB/s at $249 new is better value than either competitor on paper. In practice, software overhead partially negates the bandwidth advantage, but the gap should narrow as Intel's stack matures.

The RX 7700 XT has the best raw performance but costs $100 more. ROCm support has improved a lot since 2024, and Ollama added native AMD support. If you're willing to spend $350, the 7700 XT is the strongest 12GB option for LLM inference today.

My take: If you already own a B580 for gaming and want to try local LLMs, go for it. It works. If you're buying specifically for LLM inference, a used RTX 3060 12GB at $180 gets you there with less friction. The B580 makes sense if you want a new card that handles both gaming and LLMs at $250 and you don't mind spending an afternoon on setup.

What Runs on 12GB

With 12GB VRAM, here's what fits (see our VRAM requirements guide for the full table):

Model	Quant	VRAM	Verdict
Qwen 3.5 9B	Q4_K_M	6.6GB	Runs easily, room for 32K+ context
Qwen 3.5 9B	Q8_0	11GB	Fits tight, ~8K context
Llama 3.1 8B	Q4_K_M	5.5GB	Comfortable fit
Mistral 7B	Q4_K_M	5GB	Plenty of headroom
Qwen 2.5 Coder 14B	Q4_K_M	9.5GB	Fits, limited context
Phi-4 14B	Q4_K_M	9GB	Fits, limited context
Qwen 3.5 27B	Q4_K_M	17GB	Does not fit
Llama 3.1 70B	Any	40GB+	Does not fit

The sweet spot on 12GB is a 7-9B model at Q4_K_M. You get the model plus enough headroom for a useful context window. At Q8 you can squeeze in a 9B model but context will be tight. Anything above 14B at Q4 won't fit.

For more on what to run, check our llama.cpp vs Ollama vs vLLM comparison.

Known Issues and Workarounds

Driver versions matter a lot. Intel's GPU drivers update frequently and updates can break things. Pin a working driver version and don't update unless you have a reason to.

On newer Linux kernels, some users report "bus error (core dumped)" with Intel Arc. The IPEX-LLM team tracks these in their GitHub issues. If you hit this, try an older kernel or check for a fixed driver version. The SYCL/Level Zero backend can also fail to load model tensors with I/O errors, which is a known issue. Switching to Vulkan usually fixes both problems.

One thing that trips people up: Vulkan consistently beats SYCL on the B580. Counterintuitive, since SYCL is Intel's own stack, but if you're getting poor performance with SYCL, try Vulkan before spending hours debugging.

Ollama doesn't natively support Intel Arc the way it supports NVIDIA (CUDA) and AMD (ROCm). The IPEX-LLM bridge works but adds complexity. If your workflow depends on Ollama specifically, this is a real friction point. If you're fine with llama.cpp directly, it's a non-issue.

Flash Attention is also limited. The SYCL implementation may not work on all Arc models, and Vulkan doesn't support it at all. This affects prefill speed on long prompts but doesn't change generation speed.

Setup Checklist (Vulkan Path)

The fastest way to get running:

1. Install the latest Intel Arc GPU driver from Intel's download center
2. Download a llama.cpp release with Vulkan support
3. Download a GGUF model (start with Qwen 3.5 9B Q4_K_M from HuggingFace)
4. Run: `./llama-server -m model.gguf -ngl 99 -c 8192`
5. Open `http://localhost:8080` in your browser

That's it. No oneAPI, no conda, no environment variables. If this works and the performance is acceptable, you're done. Only go down the SYCL or IPEX-LLM path if you need something Vulkan can't provide.

If the model isn't using the GPU, check that Vulkan is detecting your Arc card: `vulkaninfo --summary` should show the B580. If it doesn't, your driver isn't installed correctly. See our Ollama not using GPU guide for general GPU detection troubleshooting, though the NVIDIA-specific fixes won't apply.

Bottom Line

The Arc B580 is a real budget option for local LLMs. 12GB VRAM at $249 with bandwidth that matches or beats the RTX 3060. The value per dollar is there.

The tax you pay is in software maturity. You'll spend more time on setup and more time in GitHub issues when something breaks. Intel's stack is improving fast, but it's not at CUDA's level and probably won't be for another year or two.

Buy it if you want a new 12GB card for gaming that also does LLM inference, and you don't mind reading GitHub issues when something breaks. Skip it if you want the smoothest possible path to running local models, in which case a used RTX 3060 12GB at $180 is the better call.

Related Guides

- VRAM Requirements for Every Local LLM
- llama.cpp vs Ollama vs vLLM: When to Use Each
- Ollama Not Using GPU: Complete Fix Guide
- What Can You Run on 12GB VRAM?
- Best GPU Under $300 for Local AI

Get notified when we publish new guides.

Subscribe — free, no spam
