

Open in app ↗



Medium



Search



Write



AI Mind

★ Member-only story

I Spent \$200 on Claude Last Month. Then I Found GLM-4.6

How Z.ai's new 355B parameter model delivers enterprise-grade coding at 1/7th the cost — and why embedded engineers like me are switching



Adham Khaled

Follow

8 min read · Oct 14, 2025



306



9



My Claude bill hit \$200 last month.

Not because I was careless. Not because I was wasting tokens on casual conversations. I was doing what every embedded systems engineer does now — letting AI help me write firmware, debug CAN protocol implementations, and architect Qt applications faster.

But \$200 a month? That's \$2,400 a year. For a freelancer juggling Fiverr gigs and personal projects, that's real money.

Then on September 29, 2025, Z.ai quietly dropped GLM-4.6. And everything changed.



The Numbers That Made Me Stop Scrolling

82.8 on LiveCodeBench versus Claude's similar performance.

200,000 token context window — enough to feed it an entire embedded Linux codebase in one go.

\$0.60 per million input tokens. Compare that to Claude's pricing.

I read those specs three times. This couldn't be real. A 355-billion parameter Mixture of Experts model delivering Claude-level performance at one-seventh the cost?

I had to test it myself.

My First Real Test: Embedded Linux Driver Code

I gave both models the same brutal task: Generate a Linux kernel module for interfacing with a custom CAN controller over SPI, complete with device tree bindings and interrupt handling.

Claude wrote beautiful code. Clean abstractions. Proper error handling. Production-ready comments.

GLM-4.6 wrote beautiful code too. But here's what stunned me — it used 15% fewer tokens than Claude for equivalent output quality.

Same logical organization. Same syntactic correctness. Same thoughtful architecture.

But cheaper. Noticeably cheaper.

What Makes GLM-4.6 Different (The Technical Deep Dive)

Let me break down what Z.ai (Zhipu AI) actually built here, because the architecture matters:

The Brain: 355B Parameter MoE Architecture

Unlike monolithic models, GLM-4.6 uses Mixture of Experts (MoE) — think of it as having 355 billion parameters but only activating the relevant “experts” for each task. This is why it's fast despite its massive size.

When I'm writing C code, it activates coding experts. When I'm debugging, it switches to reasoning experts. When I'm working with 100K+ token contexts

(like feeding it entire RTOS documentation), it efficiently manages that 200K context window without choking.

The Coding Chops: Real Benchmarks

Let's talk numbers that matter:

- LiveCodeBench: 82.8 (up from 63.3 in GLM-4.5)
- SWE-bench: 68.0 (up from 64.2 in GLM-4.5)
- AIME 25: 93.9 (98.6 with tool use)

In head-to-head real-world tests, GLM-4.6 beat Claude Sonnet 4 in 48.6% of coding scenarios and crushed DeepSeek V3.1 at 64.9%.

That's not marketing fluff. That's Z.ai's internal CC-Bench where human evaluators worked with models in isolated Docker containers across front-end dev, tool building, data analysis, and algorithms.

The Context Beast: 200K Tokens

This is where embedded engineers should pay attention. The context window jumped from 128K to 200K tokens.

You know what that means? I can feed GLM-4.6:

- Entire FreeRTOS or Zephyr RTOS documentation
- Complete Qt application source trees
- Multi-file embedded projects with hardware abstraction layers

- Legal contracts for client work
- Research papers for TinyML model optimization

All in one conversation. No context switching. No “please summarize the previous conversation” dance.

How to Actually Use GLM-4.6 (3 Methods That Work)

Let me save you the hours I spent figuring this out.

Method 1: The Cloud API (Fastest Setup)

If you just want to start coding immediately, use Z.ai’s API:

```
import requests
import json

url = "https://api.z.ai/api/paas/v4/chat/completions"
payload = {
    "model": "glm-4.6",
    "messages": [
        {"role": "user", "content": "Write a FreeRTOS task for reading CAN messa
    ],
    "max_tokens": 2000,
    "temperature": 1.0,
    "top_p": 0.95,
    "top_k": 40
}
headers = {
    "Authorization": "Bearer your-api-key",
    "Content-Type": "application/json"
}
response = requests.post(url, data=json.dumps(payload), headers=headers)
print(response.json()["choices"][0]["message"]["content"])
```

Get your API key from z.ai, and you're running. That's it.

Pro tip: Use `temperature: 1.0` and `top_p: 0.95` for coding tasks — Z.ai's official recommendations based on extensive testing.

Method 2: Local Deployment with Ollama (Privacy + Offline)

This is what I use for client projects with NDA requirements:

```
# Install Ollama
curl -fsSL https://ollama.com/install.sh | sh

# Run GLM-4.6 (the compressed version that fits on consumer hardware)
OLLAMA_MODELS=unsloth ollama serve &
OLLAMA_MODELS=unsloth ollama run hf.co/unsloth/GLM-4.6-GGUF:TQ1_0
```

The Unsloth team created optimized quantized versions. The 2-bit dynamic quantization (UD-Q2_K_XL) is only 135GB — that's a 75% reduction from the full 400GB model.

I run this on my workstation with a single 24GB GPU and 128GB RAM. The MoE layers offload to CPU, and it's plenty fast for my embedded development workflow.

Method 3: Integration with Coding Agents

Here's where it gets really powerful. GLM-4.6 works with the coding agents you already use:

For Cursor (Claude Code):

Update your `~/.claude/settings.json`:

```
{  
  "model": "glm-4.6",  
  "provider": "z.ai"  
}
```

For Windsurf, Kilo Code, Cline, Roo Code:

All support Z.ai integration. Select Z.AI as your API provider and choose GLM-4.6 from the model dropdown.

I use Cursor for embedded C/C++ work and Windsurf for Qt/QML front-ends. Having GLM-4.6 as the backend saves me roughly \$150/month compared to my old Claude-only setup.

The Hardware Reality Check

Let's be honest about what you need to run this locally:

If you want the full 355B parameter model: 400GB disk space. Good luck fitting that on consumer hardware.

If you're practical like me: Use the 2-bit quantized version (135GB). You'll need:

- 1x24GB GPU (like an RTX 3090 or 4090)
- 128GB RAM minimum

- The `-ot ".ffn_.*_exps.=CPU"` flag to offload MoE experts to CPU

Budget option: 4-bit quants work on 1x40GB GPU with MoE offloading and about 205GB total RAM.

For most embedded engineers, the API route makes more sense unless you have strict data residency requirements.

What I Use GLM-4.6 For (Real Examples)

Embedded C/C++ Development

I feed it hardware datasheets and ask for peripheral drivers. Last week, it wrote a complete STM32 UART DMA driver with circular buffer implementation in under 60 seconds.

Qt/QML Application Architecture

The 200K context window means I can give it my entire Qt project structure and ask for refactoring suggestions. It sees the big picture.

TinyML Model Optimization

When I'm deploying TensorFlow Lite models to embedded devices, GLM-4.6 helps me quantize and optimize for resource-constrained targets. It understands both the ML side and the embedded constraints.

Technical Writing

I write Medium articles about AI tools and embedded systems. GLM-4.6 helps me structure technical content and translate complex concepts for different audiences.

Debugging RTOS Issues

Feed it a FreeRTOS stack trace with your task implementations, and it'll spot race conditions and priority inversion issues faster than I can.

The Gotchas (What They Don't Tell You)

It's Not Perfect at Everything

GLM-4.6 still lags behind Claude Sonnet 4.5 on certain specialized coding benchmarks. If you're doing cutting-edge ML research or need absolute top-tier performance, Claude might still edge it out.

But for 90% of real-world embedded and software development? The difference is negligible.

The Quantized Models Need Tuning

When running locally with llama.cpp, you **MUST** use the `--jinja` flag or the output will be garbled. I spent 3 hours debugging this before reading the docs properly.

Also, for long contexts (approaching that 200K limit), enable KV cache quantization or you'll run out of VRAM:

```
./llama.cpp/llama-cli \  
--model GLM-4.6-UD-Q2_K_XL.gguf \  
--cache-type-k q4_0 \  
--jinja \  
--ctx-size 200000
```

You Need Medium Followers for API Credits

Z.ai offers a \$3/month “GLM Coding Plan” but the free tier is limited. Plan your usage accordingly.

The Real Comparison: Claude vs GPT-4 vs GLM-4.6

Let me be brutally honest about where each model wins:

Claude Sonnet 4.5: Still the king for absolute cutting-edge performance and nuanced reasoning. But expensive.

GPT-4: Broader general knowledge, better at creative tasks. Not specialized for coding.

GLM-4.6: Best price-to-performance ratio for coding. Excellent for embedded systems, firmware, and full-stack development. Massive context window. Open-source weights available.

For freelancers and indie developers? GLM-4.6 is the obvious choice.

For enterprises with unlimited budgets? Maybe you stick with Claude.

Why Embedded Engineers Should Care

Our field has a unique set of constraints:

- We work with massive codebases (RTOS kernels, HALs, protocol stacks)
- We need to understand both high-level architecture and register-level hardware

- We often work under NDAs requiring local deployment
- We're cost-conscious because embedded consulting isn't FAANG money

GLM-4.6 checks every box:

- ✓ 200K context for entire codebases
- ✓ Strong C/C++ generation and reasoning
- ✓ Local deployment options for data privacy
- ✓ 1/7th the cost of comparable cloud models
- ✓ Open-source weights (MIT license) for customization

The Future: What's Coming

Z.ai released GLM-4.6 with MIT license. That means:

- Community fine-tunes for specialized domains (embedded systems, automotive, IoT)
- Better integration with niche coding agents
- Optimized quantizations for even lower memory footprint
- Potential multimodal extensions (imagine feeding it oscilloscope captures or PCB schematics)

The embedded AI community is just starting to realize what's possible here.

How to Get Started Today (Action Steps)

If you want to test quickly:

1. Sign up at z.ai

2. Get an API key
3. Use the Python code snippet above
4. Try it with one of your real coding problems

If you want local deployment:

1. Install Ollama: `curl -fsSL https://ollama.com/install.sh | sh`
2. Pull the model: `OLLAMA_MODELS=unsloth ollama run hf.co/unsloth/GLM-4.6-GGUF:TQ1_0`
3. Start coding with full privacy

If you want coding agent integration:

1. Open Kilo Code settings
2. Add Z.ai as provider
3. Select GLM-4.6 model
4. Watch your monthly AI bill drop

The Bottom Line

I'm not switching away from Claude entirely. For the hardest, most nuanced architectural decisions, I still reach for Claude Sonnet 4.5.

But for 90% of my embedded development, firmware writing, Qt applications, and technical content? GLM-4.6 delivers comparable results at 1/7th the cost.

My September AI bill: \$200.

After switching to GLM-4.6, About \$1,800+ can be saved per year. Money I can invest in better hardware, more training, or just keeping more of what I earn as a freelancer.

Z.ai built something special with GLM-4.6 — a genuinely competitive alternative to the closed-source giants, available as both a cloud API and open-source weights.

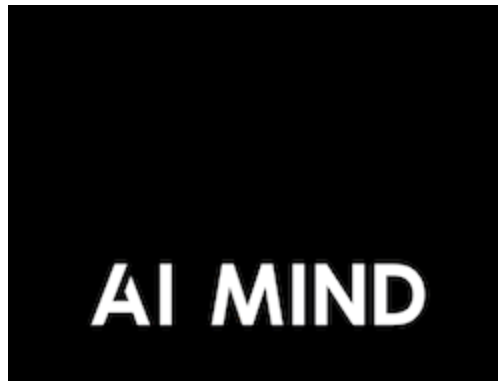
For embedded systems engineers, IoT developers, firmware hackers, and anyone writing code for resource-constrained devices, this is the model we've been waiting for.

Try it. Test it against your current workflow. Calculate your savings.

Then come back and tell me if you switched too.

If this helped you, clap 🙌 (it really helps the Medium algorithm show this to more embedded engineers). And follow me for more deep dives on AI tools, embedded systems, and making bleeding-edge tech actually useful for working engineers.

A Message from AI Mind



Thanks for being a part of our community! Before you go:

- 🖐️ Clap for the story and follow the author 🙌
- 📰 View more content in the [AI Mind Publication](#)
- 🧠 Improve your [AI prompts effortlessly and FREE](#)
- 🧰 Discover [Intuitive AI Tools](#)

Artificial Intelligence

Coding

Embedded Systems

Software Development

Software Engineering



Published in AI Mind

4.8K followers · Last published 1 day ago

Follow

Learn, explore, or build the future of AI with top stories on the latest trends, tools, and technology. Share your crazy success stories or AI-fueled fails in a supportive community.



Written by Adham Khaled

159 followers · 90 following

Follow



Embedded Systems Engineer || AI & Tech enthusiast ||
<https://linktr.ee/adhamhidawy>

Responses (9)



Alex Mylnikov

What are your thoughts?



Koushik Srikakolapu

4 days ago



Thanks for putting up so clear !



7



1 reply

[Reply](#)



Paul Proctor

2 days ago



2 bit quants on any model is a huge quality reduction. If you can afford some more ram, you might try a 4 bit xs or nl quant. Slower, but a huge quality bump.



5



1 reply

[Reply](#)



Jochen Häberle

4 days ago



in Kilo-Code, you can use GLM-4.6 via the built in Kilo Provider. No need to add another provider. This means OpenRouter is used by kilo to access the model.



7



1 reply

[Reply](#)

See all responses

More from Adham Khaled and AI Mind

:2510.01171v3 [cs.CL] 10 Oct 2025

ABSTRACT

Post-training alignment often reduces LLM diversity, leading to a phenomenon known as *mode collapse*. Unlike prior work that attributes this effect to algorithmic limitations, we identify a fundamental, pervasive data-level driver: *typicality bias* in preference data, whereby annotators systematically favor familiar text as a result of well-established findings in cognitive psychology. We formalize this bias theoretically, verify it on preference datasets empirically, and show that it plays a central role in mode collapse. Motivated by this analysis, we introduce *Verbalized Sampling* (VS), a simple, training-free prompting strategy to circumvent mode collapse. VS prompts the model to verbalize a probability distribution over a set of responses (e.g., “Generate 5 jokes about coffee and their corresponding probabilities”). Comprehensive experiments show that VS significantly improves performance across creative writing (poems, stories, jokes), dialogue simulation, open-ended QA, and synthetic data generation, without sacrificing factual accuracy and safety. For instance, in creative writing, VS increases diversity by 1.6-2.1x over direct prompting. We further observe an emergent trend that more capable models benefit more from VS. In sum, our work provides a new data-centric perspective on mode collapse and a practical inference-time remedy that helps unlock pre-trained generative diversity.

Problem: Typicality Bias Causes Mode Collapse

Solution: Verbalized Sampling (VS) Mitigates Mode Collapse

 In Generative AI by Adham Khaled

Stanford Just Killed Prompt Engineering With 8 Words (And I...

ChatGPT keeps giving you the same boring response? This new technique unlocks 2x...

 4d ago  536  12  



 In AI Mi... by TechToFit - Master Your Life with Te...




 In AI Mind by Francesco Franco

Synthetic Data: What It Is and How to Use It

(Updated 10/16/25)

Oct 8  286  4  



 In Generative AI by Adham Khaled

I Tested Google's New AI That Controls Your Computer

Google just dropped something that feels like it's straight out of science fiction, a brand ne...

★ Oct 13 🖱️ 69



Your AI Coding Assistant Is Wasting 40% of Your Tokens. Her...

DeepContext brings semantic search to large codebases, slashing token costs and makin...

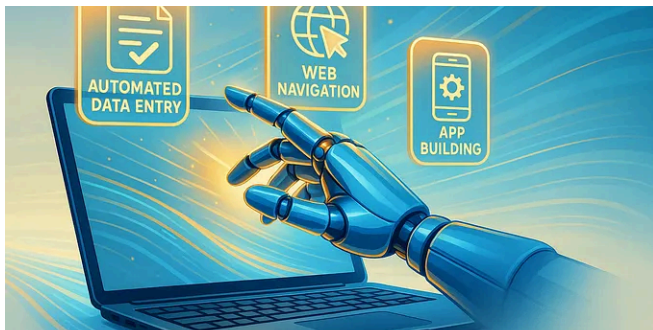
★ 3d ago 🖱️ 69 💬 2



See all from Adham Khaled

See all from AI Mind

Recommended from Medium



In CodeToDe... by TechToFit - Master Your Life wi...

I Tried Google's New AI Agents. It's a Gold Rush.

I spend my days deep in the world of AI, but every so often, something drops that makes...

★ Oct 10 🖱️ 164 💬 3



In Dare To Be Better by Max Petrusenko

Claude Skills: The \$3 Automation Secret That's Making Enterprise...

How a simple folder is replacing \$50K consultants and saving companies literal da...

★ 6d ago 🖱️ 213 💬 4



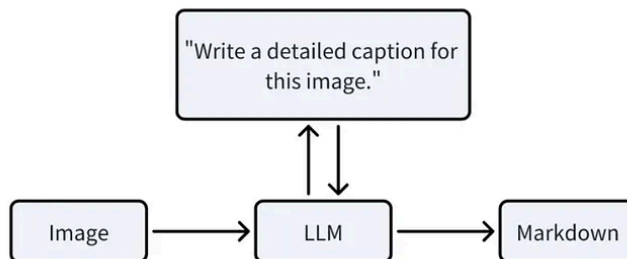



 In Towards Deep Learning by Sumit Pandey

Why Everyone Will Want DGX Spark on Their Desk—Yes,...

I just saw this picture today and was amazed, I've been waiting for this moment for a long...

★ Oct 14 🖱️ 23 💬 7  ⋮

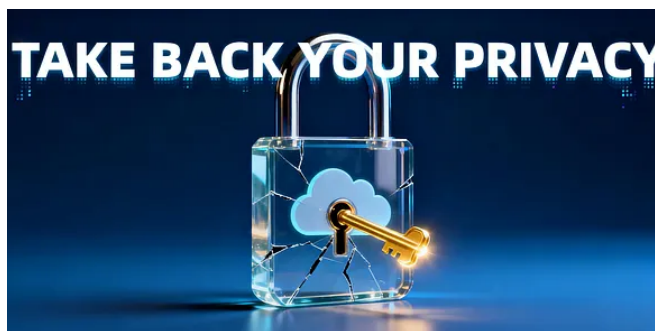


 In Coding Nexus by Civil Learning

MarkItDown: Convert Anything into Markdown—the Smart Way to Fe...

You know that feeling when you're trying to feed a PDF or a Word document into an LLM,...

★ Oct 15 🖱️ 144 💬 3  ⋮

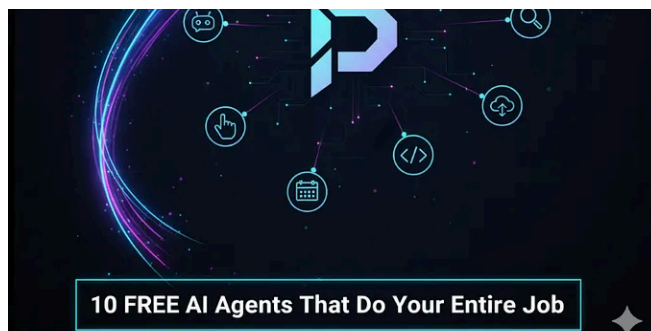



 Bytefer

The Self-Hosted Alternative to Google Notebook LM Is Here.

Google Is Reading Your Notes. Here's the 2-Minute Fix to Take Back Your Privacy.

★ 4d ago 🖱️ 141 💬 3  ⋮



 aakash

Perplexity Just Unleashed 10 FREE AI Agents That Do Your Entire Job...

Stop what you are doing. The era of the simple AI search engine is officially over.

★ Oct 7 🖱️ 708 💬 25  ⋮

[See more recommendations](#)