

Seu copiloto de código, 100 % local

Capítulo 13 — Quando dá errado: diagnosticar e corrigir (trecho) · v2026.07 · qualllm.com/copilote-local

Capítulo 13 — Quando dá errado: diagnosticar e corrigir

Na semana 1 de uma configuração local, o problema quase nunca é «o modelo é ruim». São cinco falhas, sempre as mesmas. Elas fazem você acreditar que o ambiente local não funciona. Na realidade, faltam apenas duas linhas de configuração. Este capítulo as diagnostica **por sintoma**: você parte do que vê (uma mensagem de erro, uma vazão que despenca, uma autocompletação lenta), chega à causa e, depois, à correção. Este é o capítulo de referência para diagnóstico. Os capítulos 4, 6 e 12 encaminham você para cá assim que uma falha se manifesta.

Ponto de referência: você está rodando na sua `monprojet` (a pequena API web Flask/Python que serve de fio condutor desde o capítulo 7), backend Ollama, referência de hardware RTX 5070 Ti de 16 GB. Sempre apresento a versão «para a SUA VRAM». Um conserto válido em 24 GB pode ser exatamente o reflexo errado em 8 GB.

✅ **PARA LEMBRAR — A regra de ouro da depuração local.** 90 % das falhas da semana 1 vêm de **um único número**: `num_ctx`. O `num_ctx` é o tamanho da memória de trabalho do modelo. Você o aumentou para fazer seu repositório caber, e isso empurrou o modelo para fora da VRAM. Antes de suspeitar de outra coisa, veja onde seu modelo é executado (`ollama ps`) e quanto você aumentou o contexto.

 No guia completo, este capítulo começa com uma árvore de decisão (sintoma → seção → solução) que cobre as 5 panes típicas da primeira semana. Este trecho gratuito contém a pane n.º 1 (OOM) e o início da n.º 2 (GPU não utilizada).

Seção A — OOM da GPU: «out of memory» ou taxa de geração despencando

O sintoma

Ele aparece de duas formas. A explícita: uma mensagem `CUDA error: out of memory` (ou o equivalente em ROCm/Metal), e o modelo se recusa a responder. A sorrateira, mais frequente e traiçoeira: **ele responde, mas a poucos tokens/segundo**. O modelo extrapolou a VRAM. Ollama transferiu parte das camadas para a CPU (isso é o offload), e sua GPU de 16 GB fica vendo a CPU se arrastar.

A causa

O orçamento de VRAM é simples, e você mesmo pode calculá-lo:

```
VRAM necessária ≈ (pesos do modelo quantizado)
                  + (cache KV, que cresce com num_ctx)
                  + overhead
```

O cache KV é a memória onde o modelo armazena o contexto já processado. Nos seus 16 GB, o modelo agentic padrão — Devstral Small 24B em `q4_K_M` (~14 GB) — cabe **desde que você não aumente `num_ctx`**. A armadilha clássica: você passa `num_ctx` de 8192 para 32768 para incluir mais da sua `monprojet`. O cache KV dobra ou triplica, e você ultrapassa a marca de 16 GB sem mensagem de erro — apenas o offload silencioso.

O diagnóstico em um comando

```
ollama ps
```

Leia a coluna `PROCESSOR` (a interpretação detalhada está no **capítulo 4**). Ela é o instrumento central de diagnóstico de todo o capítulo. Veja como usá-la para o OOM:

Exibição <code>ollama ps</code>	Interpretação	Ação
<code>100% GPU</code>	Tudo está na VRAM, está saudável	Nada a fazer quanto ao OOM
<code>58%/42% CPU/GPU</code> (ou qualquer %CPU diferente de zero)	Offload parcial → taxa de geração despencando	Reduzir <code>num_ctx</code> ou o tamanho do modelo
<code>100% CPU</code>	A GPU não está sendo usada de forma alguma	Vá para a Seção B (não é um OOM)

Tabela 13.1 — Leitura de `ollama ps` para diagnosticar o OOM.

O conserto (em ordem, do menos destrutivo ao mais destrutivo)

1. Reduza `num_ctx` primeiro. É a alavanca nº 1 e a menos custosa. Volte a um valor que caiba (8192 é o padrão de chat/agente em 16 GB). Verifique com `ollama ps` e depois aumente em etapas até pouco antes do offload.

```
# Teste direto: imponha um contexto modesto e veja se voltamos a 100% GPU
OLLAMA_CONTEXT_LENGTH=8192 ollama run devstral:24b
```

No Modelfile (consulte o **capítulo 11** para ver a estrutura completa), a mesma coisa de forma fixa:

```
FROM devstral:24b
PARAMETER num_ctx 8192
```

2. Desça um nível de quantização ou de tamanho. Se ainda transbordar com um `num_ctx` razoável, seu modelo é grande demais para sua placa. Fique em `q4_K_M` (o compromisso padrão entre qualidade e espaço), sem usar algo mais exigente. O ajuste fino de temp/top-p/ `num_ctx` é tratado no **capítulo 12**; a escolha do modelo certo para sua VRAM, no **capítulo 5**.

3. Limite explicitamente as camadas da GPU (caso extremo). Você quer forçar uma divisão controlada em vez de sofrer com o offload automático? `num_gpu` define o número de camadas enviadas para a GPU:

```
FROM devstral:24b
PARAMETER num_ctx 8192
PARAMETER num_gpu 99  # "o máximo possível" ; reduza este número se quiser descarregar voluntariamente para a CPU
```

⚠ AVISO — `num_gpu` não é uma fórmula mágica. Reduzi-lo não «libera» desempenho. Pelo contrário: ele envia *mais* camadas para a CPU, portanto deixa tudo mais lento. É uma ferramenta para os casos em que você quer executar voluntariamente um modelo grande demais, aceitando a lentidão. Não é uma configuração para o dia a dia. No uso normal, deixe Ollama gerenciar isso. Prefira mexer em `num_ctx` e no tamanho do modelo.

Para a SUA VRAM:

VRAM	Principal reflexo diante de OOM
8 GB	O tier agentic denso de 24B está fora do alcance; prefira um modelo leve (Qwen2.5-Coder 7B) e <code>num_ctx</code> ≤ 8192. Qualquer aumento de contexto custa caro.
12 GB (RTX 5070, RTX 4070)	Devstral Small 24B NÃO cabe aqui: escolha Qwen2.5-Coder 14B (~9 GB) ou DeepSeek-Coder-V2-Lite 16B (~10 GB) em <code>q4_K_M</code> , <code>num_ctx</code> 4096. Acima disso, monitore <code>ollama ps</code> .
16 GB (REFERÊNCIA — RTX 5070 Ti / 4080 / 5080 / RX 9070 XT, Mac M4 24 GB)	Devstral Small 24B <code>q4_K_M</code> (~14 GB) + <code>num_ctx</code> 8192 = ponto ideal agentic. Também há margem para Qwen2.5-Coder 14B em Q8.
24 GB (RTX 3090/4090, RX 7900 XTX, Mac M4 Pro 48 GB)	Você pode usar Qwen2.5-Coder 32B <code>q4_K_M</code> (~19 GB), Devstral em Q5/Q6 ou contexto longo sem offload; o OOM se torna raro.
Apple Silicon (Metal)	A VRAM é compartilhada com a RAM do sistema (memória unificada). Não há OOM brusco, mas há pressão de memória que faz swap: monitore a pressão de memória do sistema e feche processos pesados antes de uma sessão longa.

Tabela 13.2 — Reflexo diante de OOM por nível de VRAM.

Seção B — A GPU não está sendo usada de forma alguma

O sintoma

`ollama ps` exibe `100% CPU`, o ventilador da GPU está parado e a geração é lenta em todos os lugares — não despenca como em um offload, apenas fica lenta *em todos os lugares*, desde o primeiro token. Este é o caso mais frustrante: o hardware está presente, mas não é utilizado.

A causa

Ollama não detecta seu acelerador. Três causas comuns: - O runtime da GPU não foi detectado (drivers CUDA / ROCm ausentes ou antigos demais). - No Linux, Ollama está sendo executado em um contexto (contêiner, serviço systemd) que não tem acesso ao dispositivo da GPU. - No Windows/WSL, o passthrough da GPU não está configurado.

O diagnóstico

```
# NVIDIA (sua 5070 Ti) – a GPU deve aparecer e listar um processo ollama durante uma geração
nvidia-smi

# AMD ROCm
rocm-smi

# E, principalmente: leia os logs do Ollama na inicialização, quando ele informa o que detectou
```

No Linux, os logs do serviço:

```
journalctl -u ollama --no-pager | grep -i -E "gpu|cuda|rocm|library"
```

Você procura uma linha em que Ollama diga ter encontrado uma biblioteca de GPU (CUDA/ROCm). Se ele informar que mudou para CPU, você encontrou a causa. (A instalação correta do runtime de acordo com o seu sistema operacional é abordada no **capítulo 4** — se esta seção se aplica a você, muitas vezes é porque a etapa da GPU do cap. 4 foi pulada.)

O trecho termina aqui. O restante da seção B e as seções C (estouro de contexto), D (tag errada) e E (autocompletar FIM lento), além da tabela de referência sintoma → causa → solução, estão no guia.

A continuação está no guia

25 capítulos (cerca de 175 páginas) e o pacote de configurações prontas para colar: Modelfiles ajustados, Cline, Aider e Tabby, tokens FIM corretos, guia rápido VRAM → modelo → comando. Pagamento único de 24 €, atualizações vitalícias.

<https://quallm.com/copilote-local/>