本地 AI 工作站搭建与极限调优报告

基于 RTX 5070 Ti 与 9800X3D 的 128K 上下文混合推理实践

AMD Ryzen 9 9800X3D | NVIDIA RTX 5070 Ti (16GB) | Ubuntu 24.04 LTS | Hermes v0.16.0 (pipx)

一、 最终运行指标 (The Sweet Spot)

经过多轮极限调优，系统达到了硬件利用率与稳定性的完美平衡，成功支撑起 Hermes Agent 的 128K 超长上下文需求。

指标	实测数据	状态评估
显存占用 (VRAM)	**~14.0 GB / 16.3 GB**	模型加载后的稳态占用，留出 2GB+ 余量防 OOM
生成速度 (Speed)	**14 - 16 tokens/s**	远超人类阅读速度，交互极其流畅
上下文窗口	**131,072 Tokens**	满足 Hermes Agent 最严苛的长文本要求

GPU 温度 (负载/空闲)	56°C (负载) / 35°C (空闲)	散热极佳，显卡从容不迫
GPU 功耗 (负载/空闲)	~181W (负载) / ~17W (空闲)	远低于 300W TDP，功耗控制优秀

🛠

二、 环境搭建与编译踩坑

1. CUDA 版本陷阱

RTX 5070 Ti 属于 Blackwell 架构 (Compute Capability 12.0)，Ubuntu 默认源的旧版 CUDA 无法识别 `compute_120a`。

报错： `nvcc fatal : Unsupported gpu architecture 'compute_120a'`

2. 破局方案

必须手动安装 NVIDIA 官方 **CUDA 12.8.2**，并在 CMake 中**显式指定编译器绝对路径**，绕过系统的软链接干扰。最终 NVIDIA 驱动版本为 **595.71.05**。

```
cmake .. \
  -DLLAMA_CUDA=ON \
  -DLLAMA_NATIVE=ON \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_CUDA_COMPILER=/usr/local/cuda-12.8/bin/nvcc
```

三、 核心调优：显存与上下文的极限平衡

在 16GB 显存下运行 27B 模型并开启 128K 上下文，是一场"拆东墙补西墙"的艺术。

1. KV Cache 极限压缩 (保命神技)

默认 f16 精度的 KV Cache 会瞬间撑爆显存。启用 `q4_0` 量化后，128K 上下文的显存占用从 8GB 骤降至约 2.5GB。

```
--cache-type-k q4_0 --cache-type-v q4_0
```

2. CPU/GPU 混合推理 (发挥 9800X3D 优势)

为了腾出显存给 128K 上下文，我们将部分模型层 (`-ngl 50`) 卸载到 CPU。普通 CPU 此举会导致速度暴跌，但 **9800X3D 拥有 96MB 超大 L3 缓存**，能完美吞下这些层的权重，实现降维打击。

3. 线程数与 Batch Size 的精调

9800X3D 为 8核16线程，推理线程 `-t 8` 绑定物理核最佳。 `-b 512` 牺牲微小首字延迟，换取长文本输入时的显存峰值锁定，确保 100% 稳定不 OOM。

四、 Hermes Agent 部署与协同

1. 部署方式

Hermes Agent **v0.16.0** 通过 `pipx` 安装，CLI 模式运行。

2. 无缝对接本地 API

通过 Hermes 配置将 Endpoint 指向本地 `http://127.0.0.1:58080/v1`，并强制覆写 Context Length 为 `131072`，成功绕过 64K 最低门槛限制。

3. 终极形态：Task Delegation (任务委派)

成功配置了"云端大脑 + 本地双手"的协同模式。云端模型（DeepSeek v4 Flash）负责复杂逻辑规划，本地 Qwen3.6-27B 负责执行 Terminal / File 操作，实现了**零云端执行成本、绝对数据隐私**的 Agent 体验。

> **委派架构：**
> 主模型：DeepSeek v4 Flash (云端，负责推理/规划)
> 子 Agent：Qwen3.6-27B-Q3_K_M (本地 127.0.0.1:58080，负责执行)

五、 破除 AI 幻觉：确立终极参数

在调优过程中，Hermes Agent 自身给出了优化建议，但包含了致命的"硬件幻觉"和"显存误判"。

> **Hermes 的错误建议：**
> 1. 建议 `-t 12` (错误认为 9800X3D 是 12核)。
> 2. 建议 `-b 2048` (误以为只吃内存，实则会导致 GPU 计算缓冲区峰值暴涨，引发 OOM 闪退)。

> **最终修正：**
> 1. `-t 8`：9800X3D 为 8核16线程，LLM 推理应绑定物理核，消除超线程缓存竞争。

2. **保持 `-b 512`**：牺牲微小的首字速度，彻底锁死长文本输入时的显存峰值。

六、 最终部署清单

底层启动命令 (llama-server 核心参数)

```
llama-server \
  -m "$HOME/Downloads/Qwen3.6-27B-Q3_K_M.gguf" \
  -ngl 50 \
  -c 131072 \
  -fa on \
  --cache-type-k q4_0 \
  --cache-type-v q4_0 \
  -t 8 \
  -b 512 \
  --port 58080 \
  --host 127.0.0.1
```

可用脚本文件

脚本路径	用途
`~/.local/bin/run-llama-terminal.sh`	开机自启动：gnome-terminal 弹窗运行 llama-server (sleep 3 + 前台 + | tee 日志)
`~/.local/bin/run-hermes-terminal.sh`	开机自启动：gnome-terminal 弹窗运行 Hermes CLI (sleep 12 + curl 健康检查 + exec)
`~/.local/bin/runllama`	(旧) 手动启动 llama-server 的原始脚本，保留作参考

`~/rescue-hermes.sh`	应急脚本：无弹窗后台启动，适合 SSH 远程恢复场景
`~/backup-hermes.sh`	备份脚本：备份 Hermes 配置、模型缓存、系统信息到 U 盘

七、 开机自启动方案

最终采用 **Gnome Autostart + 双 gnome-terminal 弹窗** 方案，实现开机后自动拉起 llama-server 和 Hermes Agent CLI。

> **架构：**
> GNOME 会话管理器 → 2 个 .desktop 文件 → 2 个 gnome-terminal 窗口 → 2 个 shell 脚本

Desktop 文件

`~/.config/autostart/llama-server.desktop`

```
[Desktop Entry]
Type=Application
Name=llama-server
Comment=本地 LLM 服务
Exec=gnome-terminal --title="llama-server" --geometry=120x40 -- /home/kevonlee/.loca
Terminal=false
StartupNotify=true
Categories=Development;
X-GNOME-Autostart-enabled=true
```

`~/.config/autostart/hermes-agent.desktop`

```
[Desktop Entry]
Type=Application
Name=Hermes Agent
Comment=AI 助手 CLI
Exec=gnome-terminal --title="Hermes Agent" --geometry=100x30 -- /home/kevonlee/.loca
Terminal=false
StartupNotify=true
Categories=Development;
X-GNOME-Autostart-enabled=true
```

Shell 脚本

`~/.local/bin/run-llama-terminal.sh`：

```
#!/bin/bash
sleep 3
MODEL_PATH="$HOME/Downloads/Qwen3.6-27B-Q3_K_M.gguf"
SERVER_BIN="$HOME/llama.cpp/build/bin/llama-server"
LOG_FILE="$HOME/.hermes/logs/llama-server.log"
mkdir -p "$(dirname "$LOG_FILE")"
echo "  llama-server 启动中..."
echo "  模型: Qwen3.6-27B-Q3_K_M.gguf"
echo "  配置: -ngl 50 -c 131072 | 端口 58080"
echo "  API:  http://127.0.0.1:58080/v1"
"$SERVER_BIN" \
  -m "$MODEL_PATH" \
  -ngl 50 -c 131072 -fa on \
  --cache-type-k q4_0 --cache-type-v q4_0 \
  -t 8 -b 512 --port 58080 --host 127.0.0.1 \
  2>&1 | tee "$LOG_FILE"
echo "  llama-server 已退出 (exit=$?)"
read
```

`~/.local/bin/run-hermes-terminal.sh`：

```
#!/bin/bash
sleep 12
LOG_FILE="$HOME/.hermes/logs/hermes-startup.log"
```

```
mkdir -p "$(dirname "$LOG_FILE")"
echo "等待 llama-server 加载模型..."
for i in $(seq 1 45); do
    if curl -s http://127.0.0.1:58080/health > /dev/null 2>&1; then
        echo " llama-server 就绪!"
        break
    fi
    if [ $i -eq 45 ]; then
        echo " 警告: llama-server 未在预期时间内就绪"
    fi
    sleep 2
done
echo "启动 Hermes Agent CLI..."
exec /home/kevonlee/.local/bin/hermes chat --cli
```

踩坑记录（为什么不能更简单）：

1. **Desktop Entry 限制**：Exec= 行不能包含 `&&`、`|`、`;` 等 shell 运算符，所有逻辑必须进脚本。
2. **gnome-terminal D-Bus 竞争死锁**：两个 .desktop 同时调用 gnome-terminal 会导致 D-Bus 激活超时，两个窗口都不弹。必须脚本内 `sleep 3` + `sleep 12` 错峰。
3. **isatty 检测**：`hermes chat --cli` 用 `isatty()` 判断终端，`| tee` 会破坏检测。所以 hermes 脚本必须用 `exec` 直连，llama-server 不涉及 isatty 可用 `| tee`。
4. **systemd --user 走不通**：开机早期 D-Bus session 未就绪。
5. **xterm 备用方案**：如 gnome-terminal 未来失效，可回退到 xterm + Noto Sans Mono CJK SC 字体。

八、 应急与维护

急救脚本 (~/rescue-hermes.sh)

当开机自启动失败或需要 SSH 远程恢复时使用：

```
# SSH 登录后执行
bash ~/rescue-hermes.sh
```

该脚本以无弹窗后台模式启动 llama-server + Hermes，适合远程终端场景。

备份脚本 (~/backup-hermes.sh)

定期备份 Hermes Agent 配置、模型缓存、技能和系统信息：

```
bash ~/backup-hermes.sh
```

支持 U 盘离线备份、指定备份路径等功能。备份目录格式：`~/hermes-backup-YYYYMMDD_HHMMSS/`

Generated by Qwen & Kevonlee | 最后更新：2026-06-11

Co-created with **Hermes Agent** (Nous Research) — 本地 AI 智能体协作实践

"在本地硅片上，构建属于自己的超级智能体。"