DGX Spark 推理部署选型与实践

在一台 128GB 小机器上，把 LLM、多模态、语音与绘图跑成生产力

DGX Spark 部署小记

2026 年 6 月

这份文档把我们在 NVIDIA DGX Spark（GB10 Grace Blackwell，128GB 统一内存）上从零部署一系列推理服务的过程做一次复盘：最终选定了哪些模型、为什么这台**小机器**适合（或不适合）某类模型、哪些是真正能当生产力用的、以及 ComfyUI 是怎么部署的。结论先放这：**NVFP4 量化 + 小激活 MoE + 投机解码（MTP），是这台机器上跑大模型的黄金组合**。

1 硬件背景：为什么它是一台「特别」的小机器

DGX Spark 不是普通的消费级机器，理解它的硬件特性是一切选型的前提：

- **SoC**：GB10 Grace Blackwell 超级芯片，GPU 算力等级 `sm_121`（Blackwell），CPU 为 **aarch64**。
- **内存**：128GB（实际可用约 121 GiB）**统一内存** LPDDR5x，CPU 与 GPU 共享，没有独立显存概念。
- **关键瓶颈**：统一内存带宽约 **273 GB/s**。LLM 解码（decode）是**内存带宽受限**的——每生成一个 token 要把模型权重读一遍。所以**权重越小、读得越快**。
- **软件栈**：DGX OS（基于 Ubuntu），CUDA 13，预装驱动 + Docker + NVIDIA Container Toolkit。aarch64 + CUDA 13 意味着**很多 wheel 和镜像必须挑 arm64 / sbsa 版本**，这是后面所有踩坑的根源。

一句话推论：在 273 GB/s 带宽下，一个 55GB 的 bf16 模型解码上限约 5 tok/s；同一个模型量化到 NVFP4（约 14GB）后上限拉到约 19 tok/s。**量化不是可选项，是必选项**。

2 什么样的模型适合 DGX Spark（选型原则）

经过十几个模型的实测，我们总结出这台机器的选型铁律：

1. **必须 NVFP4**。Blackwell 原生支持 FP4 张量核心。NVFP4（W4A4）既省带宽又用上 FP4 算力。**注意**：llama.cpp 的 GGUF Q4_K 不会用 FP4 张量核心，在这台机器上是浪费。
2. **小激活 MoE 优于 Dense**。解码速度看的是**每 token 激活的参数量**，不是总参数量。一个 80B-A3B（激活 3B）的 MoE，解码比 27B Dense 还快。
3. **4-bit 权重 $\lesssim$ 100GB**。121GB 要留给 KV 缓存和操作系统，权重压到 100GB 以内才有富余。这把 400B+ 的巨无霸（GLM-5.1、Kimi K2、DeepSeek V4）直接排除在外。

4. **投机解码（MTP）是免费加速**。带 MTP（Multi-Token Prediction）draft 的模型，单请求解码能再快 40%。

5. **引擎二选一**：源码编译的 SM121 vLLM，或 `lmsysorg/sglang:v0.5.12-cu130`（实测 SGLang 在 sm_121 上可直接用，且支持 NVFP4 MoE，这是个重要发现）。

3 选型历程：我们试过的模型（含踩坑）

下表是这台机器上实测过的主要模型，按时间顺序排列。**速度均为 NVFP4 量化后实测**：

模型	类型	单请求	结论 / 坑
Qwen3.5-122B-A10B	MoE 122B/10B 多模态	~13 tok/s	太重，独占整机（102GB GPU），不适合日常
GRM-2.6-Opus	Dense 27B VL	13.7 tok/s	自量化 NVFP4：必须排除 `*linear_attn*`，否则 GDN 递归被 FP4 破坏，输出全是空格
Qwen3-Coder-Next	80B-A3B MoE	~35 tok/s	智能体编码强（`qwen3_coder` 工具），70.6% SWE-bench，3B 激活所以快
Qwen3.6-27B	Dense 27B VL	15–20 tok/s	**代码质量最强**（77.2% SWE-bench），带 MTP，但 Dense 偏慢
Nemotron-3-Super	120B-A12B MoE	~16 tok/s	原生 NVFP4+MTP，但 Mamba+MTP 在我们的 vLLM 崩溃，无 MTP 时反而比 27B 慢，**放弃**
Nex-N2-mini	Qwen3.5-MoE VL 35B-A3B	**57 tok/s**	**全场最快**，SGLang 部署，多模态，3B 激活

两个最贵的教训：

- **自量化 NVFP4 时，别动 GDN 线性注意力层**。GRM 第一版把 `linear_attn` 投影量化进 FP4，破坏了 Gated-Delta-Net 的递归状态，模型只会吐空格。modelopt 的「假量化」自检看不出来，必须看真实 `/v1/completions` 是否为空白。
- **原生带 MTP 不等于能用上 MTP**。Nemotron 的 MTP 需要更新版 vLLM；它的 Mamba 混合层在我们的 vLLM 上和投机解码不兼容（`selective_state_update` 断言失败）。结果是「最该快的模型反而最慢」。

4 最终选型：三个主力模型

折腾一圈后，落地为三个真正会用的大模型（**内存互斥，一次只跑一个**）：

模型	定位	速度	引擎
Nex-N2-mini-NVFP4	综合主力：对话 + 多模态 + 工具 + 推理，全场最快	57 / 564 tok/s	SGLang v0.5.12
Qwen3.6-27B-NVFP4	代码与智能体质量最强，Claude Code 后端首选	15–20 / 426	vLLM（SM121）
Qwen3-Coder-Next-NVFP4	智能体编码 + 速度兼顾，原生 `qwen3_coder` 工具格式	~35	vLLM（SM121）

（速度格式：单请求 / 64 并发聚合，单位 tok/s。三者均为 256K 上下文。）

4.1 选型逻辑

- 要**快、要多模态、日常用** → **Nex-N2-mini**。Qwen3.5-MoE 只激活 3B，配 SGLang 的调度，单请求 57 tok/s、64 并发聚合 564 tok/s，是全场最快，还自带视觉。
- 要**写代码、跑 Agent** → **Qwen3.6-27B**。77.2% SWE-bench Verified，在 128GB 能装下的模型里代码质量天花板，Terminal-Bench 持平 Claude 4.5 Opus。代价是 Dense 27B 偏慢。
- 要**编码 Agent 又要快** → **Coder-Next**。80B-A3B MoE，35 tok/s，原生 compressed-tensors NVFP4 加载干净。

5 哪些是「真生产力」，哪些只是「能跑」

这是这份文档最该回答的问题。**能加载 ≠ 能当生产力**。

5.1 真生产力

- **Nex-N2-mini**：日常对话、多模态理解、批量评测（eval）。64 并发 564 tok/s 的聚合吞吐，是 eval 友好的。
- **Qwen3.6-27B**：接到 Claude Code 当本地代码后端，工具调用走 `qwen3_coder` 解析器。
- **语音栈（ASR + TTS）**：Qwen3-ASR + Kokoro，组成「录音 → 识别 →LLM→ 合成 → 播放」的完整语音管线。
- **LocateAnything 视觉检测**：上传图片或开摄像头实时框选目标的 Web 应用。

5.2 只是「能跑」，不建议当主力

- **Qwen3.5-122B**：13 tok/s 且独占整机，交互体验差，只适合偶尔跑高质量单条。

- **Nemotron-3-Super**：它的卖点 MTP 在我们的栈上跑不起来（vLLM 的 Mamba+MTP 不兼容、SGLang 的 NEXTN 对它接受率 0），无 MTP 时 180 tok/s（64 并发）反而不如 27B 的 426。**卖点用不上，就不是生产力**。
- **任何 GGUF Q4**：不走 FP4 张量核心，在这台机器上是浪费算力。

6 语音与视觉栈

6.1 ASR：Qwen3-ASR-1.7B（中英识别首选）

- **选型理由**：2026-01 发布，中文（22 种方言）+ 英文 + 52 语种，比 Whisper 的中文 CER 低数倍，比老的 SenseVoice（2024）更新更准。1.7B 约 4.7GB，能与 LLM 共存。
- **部署**：官方 `qwen-asr` pip 包（transformers 后端）。基础镜像用 NGC pytorch（保留 sm_121 GPU torch），`pip install qwen-asr soundfile librosa fastapi`。
- **服务**：FastAPI 包一层，`POST /asr` 上传音频 → `{text, language}`，自动语种检测。端口 **8200**，中英实测正确。

6.2 TTS：Kokoro-82M（稳定可用的最佳替代）

- **诚实说明**：原计划用中文音质最好的 **CosyVoice2**，但它在这台 aarch64 / sm_121 / nv-torch 机器上**装不起来**——它整条链硬依赖 `torchaudio`，而 torchaudio 的 C++ 扩展无法对接 nv-torch（ABI 不匹配），matcha-tts 在 aarch64 编译失败，CosyVoice 还硬绑 torch 2.3.1。三种修法都失败。
- **结论**：**Kokoro 是无 torchaudio 依赖、稳定可用的最佳 TTS**。82M 很快，中英都不错。`pip install kokoro "misaki[zh,en]"`，纯 torch + soundfile。端口 **8300**，输出 24kHz WAV。
- **关键经验**：**这台机器上没有可用的 torchaudio**。选音频模型时，优先挑「不依赖 torchaudio」的（Kokoro、qwen-asr 用 librosa/soundfile）。

6.3 视觉：LocateAnything-3B（检测 Web 应用）

- NVIDIA 的视觉定位/检测模型（custom arch，不能上 vLLM），用 transformers 4.57.1 + `trust_remote_code` 跑。
- 做成 Web 应用：图像上传检测 + 摄像头实时检测，框（bbox）叠加到画面。端口 **8095**（HTTPS 自签，摄像头需要安全上下文）。`decord` 在 aarch64 无 wheel，用空壳 stub 顶替（只处理图像帧，视频路径不触发）。

7 ComfyUI 部署

ComfyUI 用 **docker-compose** 部署，绑定 127.0.0.1，不污染宿主。要点：

- **Manager**：装 ComfyUI-Manager，用 `--enable-manager-legacy-ui` 开老版 UI（新版按钮找不到）。外部文件夹加插件后，UI 上没有重启按钮——直接重启容器即可。

- **Civitai 接入**：配置 API key，拉 LoRA 和模型。
- **Hunyuan 视频工作流**：跑通了 HunyuanVideo 文生视频。
- **aarch64 踩坑清单**（这台机器特有）：
  - `functorch` 缺失 → 用搜索式 shim 转发到 `torch._functorch`。
  - `transformers` 必须钉 **4.50.3**（为了 `_validate_images_text_input_order` 和 GenerationMixin）。
  - `sageattention` ABI 不匹配，需对应版本。
  - ImageResizeKJ 的 off-by-one：`keep_proportion=False` + `divisible_by=16`。
  - HunyuanVideo 帧数：`num_frames` 从 89 改成 73（VAE 张量尺寸对齐）。

经验：ComfyUI 在 aarch64 上是「一路修依赖」的过程，核心矛盾都是**为 x86/CUDA12 写的轮子撞上 arm64/CUDA13**。模式和 LLM 部署一样：保留 NGC 的 GPU torch，单独修掉不兼容的小依赖。

8 推理引擎与优化要点

8.1 两套引擎

- **vLLM（源码编译 SM121）**：基于 jilycn 的 dev210 构建，叠加 torchfix（恢复 GPU torch）+ ropefix（修 transformers 5.2 的 `partial_rotary_factor` rope bug）+ fla（GDN 快核）= `vllm-spark:dev210-fast`。支持 `qwen3_5` / `qwen3_next` / `NemotronH` 等架构。
- **SGLang（v0.5.12-cu130）**：**重大发现——SGLang 在 sm_121 上开箱即用**，无需源码编译，且**支持 compressed-tensors NVFP4 MoE**（社区 issue #13639 的缺口在这个版本已修）。Nex-N2-mini 就是跑在 SGLang 上，全场最快。

8.2 把吞吐压满的「优化全家桶」

对一个模型，依次叠加这些，能从基线翻几倍：

1. **NVFP4 量化**——权重 4-bit，解码带宽直接 1/4。
2. **MTP 投机解码**——`num_speculative_tokens=2` 实测最优（=3 在高并发下因草稿计算抢占反而掉吞吐）。
3. **FP8 KV 缓存**——KV 容量翻倍，并发更高。
4. **CUDA Graphs** ——去掉 `--enforce-eager`，省内核启动开销。
5. **Prefix Caching** ——eval 共享系统提示时命中缓存。
6. **fla GDN 核**——给 qwen3_next 的线性注意力层用上 Triton 快核。

以 Qwen3.6-27B 为例，这套组合把单请求从 12 推到 **19–20 tok/s**，64 并发聚合从 394 推到 **426**，峰值 128 并发 **506 tok/s**。

9 当前服务清单

端口	服务	说明
:8000	LLM（互斥）	Nex-N2-mini / Qwen3.6-27B / Coder-Next，OpenAI 兼容
:8200	ASR	Qwen3-ASR-1.7B，`POST /asr` 音频转文字
:8300	TTS	Kokoro-82M，`POST /tts` 文字转 24kHz WAV
:8095	LocateAnything	视觉检测 Web（上传 + 摄像头实时）
:8090	spark-manager	容器管理面板
:3000	new-api	LLM 网关

服务都设了 `--restart unless-stopped` 开机自启;大模型与语音/视觉服务通过降低主模型的 `mem-fraction` 实现共存。

10 一页纸结论

- **这台小机器的甜点**：NVFP4 + 小激活 MoE + MTP + SGLang/vLLM 优化全家桶。
- **日常主力选 Nex-N2-mini**（最快、多模态）；**写代码选 Qwen3.6-27B**（质量天花板）；**要编码 Agent 的速度选 Coder-Next**。
- **不要碰**：400B+ 巨无霸（装不下）、GGUF Q4（不走 FP4 核）、卖点用不上的模型（Nemotron 的 MTP）。
- **aarch64 / CUDA13 / nv-torch 的通用解法**：保留 NGC 的 GPU torch，单独修掉为 x86/CUDA12 编译的依赖；音频模型避开 torchaudio。
- **最意外的收获**：SGLang 在 sm_121 上开箱可用且支持 NVFP4 MoE——这让最快的模型成为可能。