



SGLnag

SGLang is a high-performance open-source serving framework and domain-specific programming language designed for efficient deployment and control of large language models (LLMs) and vision-language models (VLMs). It enables low-latency, high-throughput inference across environments ranging from single GPUs to large distributed clusters, supporting advanced features and hardware acceleration. [\[1\]](#) [\[2\]](#) [\[3\]](#) [\[4\]](#) [\[5\]](#)

Core Features

- **Backend Runtime:** SGLang's backend system integrates RadixAttention for automated and efficient key-value (KV) cache reuse, resulting in faster execution during multi-turn and prefix-sharing LLM calls. It supports techniques such as continuous batching, paged attention, quantization (including FP4/FP8/INT4/AWQ/GPTQ), multi-modal batching, and various forms of parallelism (tensor, pipeline, expert, data). [\[2\]](#) [\[4\]](#) [\[5\]](#)
- **Frontend Language:** SGLang provides a Python-embedded language for programming LLM applications, offering primitives for chained generation, advanced control flow, multi-modal input handling, and structured output constraints (e.g., JSON validation via regex). This makes complex workflows more programmatically expressive and easier to extend. [\[3\]](#) [\[5\]](#)
- **Model Support:** The framework supports a broad set of generative models (LLaMA, Qwen, DeepSeek, GLM, GPT, Gemma, Mistral, and more), embeddings (e5-mistral, gte), and reward models. It is compatible with Hugging Face and OpenAI APIs. [\[2\]](#)
- **Hardware Compatibility:** SGLang runs on NVIDIA GPUs (including H100, A100, GB200), AMD GPUs, Intel Xeon CPUs, Google TPUs, and Ascend NPUs for maximum flexibility. [\[4\]](#) [\[2\]](#)
- **Serving and Deployment:** SGLang is optimized for real-time serving of AI assistants, scalable backend deployment, and multi-modal applications such as image and video captioning. [\[6\]](#) [\[4\]](#)
- **Community and Ecosystem:** SGLang is widely adopted in industry, powering over 300,000 GPUs globally and actively supported by the open-source community. [\[2\]](#)

Programming with SGLang

- SGLang supports asynchronous stream execution where generative and selection primitives are submitted for non-blocking asynchronous execution (similar to CUDA kernel launches in Python). [\[5\]](#)
- Primitive operations include generation (`gen`), option selection (`select`), prompt manipulation (`+=`), parallel (`fork/join`), and multimodal input (`image`, `video`). [\[5\]](#)

- Structured outputs can be enforced (e.g., JSON schemas via regex constraint during generation).^[5]

Typical Use Cases

- LLM-based AI assistants requiring fast, highly parallel, and interactive serving.^[4]
- Multimodal applications needing integrated vision and language workflows (e.g., captioning, video analysis).^{[6] [4]}
- Research and production environments deploying models to heterogeneous clusters with advanced quantization and batching for cost and performance efficiency.^{[4] [2]}

SGLang addresses many modern LLM application challenges by combining a flexible programming language interface with a high-efficiency backend, making it suitable for both academic research and industrial deployment.^{[3] [2] [4] [5]}

✱

1. <https://github.com/sgl-project/sglang>
2. <https://sgl-project.github.io>
3. <https://sky.cs.berkeley.edu/project/sglang/>
4. <https://rocm.blogs.amd.com/artificial-intelligence/sglang/README.html>
5. <https://openreview.net/pdf?id=VqkAKQibpq>
6. <https://lmsys.org/blog/2025-11-07-sglang-diffusion/>
7. <https://arxiv.org/abs/2312.07104>
8. <https://qwen.readthedocs.io/en/latest/deployment/sglang.html>