ChatGPT

2.2 Варианты инференса on-prem (5 мин)

Стек запуска. На практике локальное развёртывание LLM выглядит как набор сервисов: движок инференса, контейнеризация и механизмы квантования. Эта часть знакомит с популярными открытыми решениями, сравнивает их и показывает, как выбор стека влияет на стоимость, производительность и удобство.

vLLM — производительность и масштабируемость

- **Сквозная оптимизация.** Сервис vLLM построен как высокопроизводительный сервер инференса. В последних релизах реализованы *PagedAttention* для эффективного хранения KV-кеша, непрерывная пакетная обработка запросов, CUDA/HIP графы и специализированные ядра (FlashAttention/FlashInfer) — всё это даёт «state-of-the-art» пропускную способность [1].
- **Поддержка квантования.** vLLM умеет загружать модели в форматах **GPT-Q**, **AWQ**, INT4/INT8/FP8, что уменьшает требования к памяти [2].
- **Параллелизм и стриминг.** Движок поддерживает параллельные методы генерации (beam search, параллельное семплирование) и несколько схем параллелизма — tensor, pipeline, data, expert. API сервера совместимо с OpenAI и поддерживает потоковую выдачу токенов [3].
- **Аппаратная независимость.** vLLM работает на NVIDIA, AMD и Intel GPU/CPU, PowerPC и ARM, и даже TPU — полезно для гибридных кластеров [4].

Когда выбирать: vLLM рекомендуется для масштабного сервинга, где важны максимальная пропускная способность и работа на различных GPU. Благодаря батчингу и многоядерности vLLM легко масштабировать на несколько узлов. Недостаток — сложность сборки и необходимость CUDA/HIP-совместимого оборудования; для небольших локальных установок он может быть избыточным.

Hugging Face Text Generation Inference (TGI)

- **Простой запуск и наблюдаемость.** TGI предоставляет простой командный запуск для Llama, Falcon, StarCoder и др. с прометеевскими метриками и трассировкой OpenTelemetry [5], что облегчает эксплуатацию.
- **Параллельность и батчинг.** Сервис применяет тензорный параллелизм для ускорения инференса на нескольких GPU, выполняет *continuous batching* — группирует входящие запросы для лучшей загрузки GPU [6].
- **Стриминг и управление выводом.** Через SSE поддерживается потоковая выдача токенов; доступны *logits warpers* (temperature, top-k, top-p), стоп-последовательности, лог-вероятности и watermarking [7].
- **Квантование и fine-tune.** TGI поддерживает bitsandbytes/GPT-Q квантование и может служить дообученные модели. Опция *Guidance* заставляет модель генерировать структурированный вывод и вызывать функции [8].

Когда выбирать: TGI подойдёт тем, кто хочет «из коробки» поднять open-source-модели с минимальными усилиями, получая при этом мониторинг. Для больших нагрузок TGI уступает vLLM по пропускной способности, но обеспечивает простоту и богатый набор «обвязки» (события SSE, функции, лог-вероятности).

Ollama — простота развёртывания на ПК и сервере

- **CLI и многоплатформенность.** Установив Ollama на macOS, Windows или Linux, можно запустить модель одной командой: `ollama run gemma3` [9]. Эта простота делает Ollama популярным среди разработчиков, которым нужно быстро протестировать модели на ноутбуке или сервере.
- **Потоковая выдача и tool calling.** Стриминг включён по умолчанию, благодаря чему сообщения выводятся сразу. Механизм разделяет *chatting* (частичные ответы), *thinking* (вывод рассуждений) и *tool calling* — вызовы инструментов прямо из ответа [10].
- **Прозрачное мышление.** Для поддерживаемых моделей (GPT-OSS, Qwen 3, DeepSeek) можно включить поле `thinking`, чтобы отделить reasoning-трассировку от окончательного ответа; параметр `think` позволяет управлять глубиной трассы [11].
- **Аппаратная поддержка.** Ollama работает на NVIDIA GPU с вычислительной способностью ≥5.0 и поддерживает выбор GPU через `CUDA_VISIBLE_DEVICES`; при отсутствии совместимого GPU система автоматически переходит на CPU [12].

Когда выбирать: Ollama отлично подходит для прототипирования и настольных сценариев: минимум настройки, поддержка reasoning-трассы и вызова инструментов. Однако серверная нагрузка ограничена, поддержка моделей меньше, а управление батчингом и масштабированием отсутствует.

llama.cpp и GGUF — inference на ЦПУ и экзотическом железе

- **Нативный C/C++ движок.** `llama.cpp` позволяет запускать LLM-модели без внешних зависимостей. Он оптимизирован для Apple Silicon (NEON/Metal), x86 (AVX/AVX2/AVX512/AMX), RISC-V и содержит CUDA, HIP и Vulkan/SYCL бэкенды [13]. Движок поддерживает гибридный режим CPU+GPU, что позволяет частично ускорять модели, не помещающиеся в VRAM [14].
- **Глубокое квантование.** Поддерживаются форматы 1.5-, 2-, 3-, 4-, 5-, 6- и 8-битной квантизации, что резко снижает требования к памяти [15]. Появились и новые форматы (MXFP4) для ускорения.
- **GGUF — универсальный контейнер моделей.** Документ GGUF описывает формат, который хранит все метаданные модели в одном файле, обеспечивает однозначную интерпретацию и позволяет загружать модель с помощью mmap без дополнительных файлов [16]. Формат спроектирован как расширяемый и поддерживает единственный файл для удобства распространения [17].

Когда выбирать: `llama.cpp`/GGUF подходит для офлайн-инференса на ноутбуках, серверах без CUDA, а также для встраивания в C/C++-приложения. Квантованные модели в GGUF можно легко переносить между системами. Недостаток — отсутствие продвинутого батчинга и меньшее количество оптимизаций, чем у vLLM/TGI.

AutoAWQ — быстрое 4-битное квантование

- **Алгоритм AWQ.** AutoAWQ реализует *activation-aware weight quantization* — весовая квантизация, которая ускоряет модели и уменьшает объём памяти примерно в 3 раза по сравнению с FP16 [18].
- **Поддержка железа и инсталляция.** Для работы с GPU требуется NVIDIA с compute capability ≥7.5 (Turing и новее) или ROCm-совместимые AMD GPU; поддерживаются также CPU Intel/AMD via Triton [19].
- **Производительность.** Авторы рекомендуют два варианта AWQ: *GEMV* (быстрее, но только для batch size = 1) и *GEMM* (эффективна при небольших батчах) [20]. При больших батчах

модель становится compute-bound, поэтому FP16 (vLLM) обеспечивает лучшую пропускную способность [21].

Когда выбирать: AWQ полезен для размещения больших моделей на ограниченных GPU/CPU, когда важна экономия памяти. Однако квантованные модели могут требовать сложной установки и иметь ограничения по батчам; для высоких нагрузок лучше использовать FP16 с vLLM.

NVIDIA NIM — контейнеризированные микросервисы

- **Преднастроенный стек.** По сути NIM (Neural Inference Microservices) представляет собой контейнеры, включающие предварительно оптимизированные модели (DeepSeek, Llama, Mistral, SDXL и др.), **полный рантайм-стек** (TensorRT-LLM, CUDA, Triton) и API (REST/gRPC/OpenAI-совместимый) [22].
- **Сокращение DevOps-барьера.** NIM упрощает развёртывание: микросервисы поставляются как готовые контейнеры, что позволяет перейти от модели к рабочему сервису за минуты, без сложной настройки CUDA, Triton и Kubernetes [23].
- **Контейнеризация и масштабирование.** Каждая модель запускается как микросервис; благодаря этому легко масштабировать и изолировать модели, а инфраструктурные решения вроде GreenNode обеспечивают динамическое распределение GPU, мониторинг и масштабирование [24].

Когда выбирать: NIM подходит крупным организациям, которым нужна готовая инфраструктура с минимальной DevOps-нагрузкой и лицензированными NVIDIA-моделями. Преимущества — низкая латентность, простота масштабирования и поддержка коммерческих моделей; недостаток — платный доступ и привязка к экосистеме NVIDIA.

Triton Inference Server — универсальный сервер для всех фреймворков

- **Гибкость фреймворков.** Triton Inference Server — это открытый сервер, который позволяет развернуть модели из TensorRT, PyTorch, ONNX, OpenVINO, RAPIDS FIL и других ML-фреймворков [25].
- **Поддержка разных архитектур.** Он выполняет инференс на NVIDIA GPU, x86/ARM CPU и AWS Inferentia, работает в облаке, центре обработки данных и на периферии [26].
- **Продвинутые функции.** Triton обеспечивает одновременное выполнение нескольких моделей, динамический и последовательный батчинг, сборки в виде ансамблей или бизнес-логики, HTTP/REST и gRPC протоколы, а также API на C/Java для встраивания [27]. Метрики GPU-использования, пропускной способности и задержки помогают мониторить производительность [28].

Когда выбирать: Triton — это «швейцарский нож» для инференса, поддерживающий широкий спектр фреймворков и устройств. Подходит для гетерогенных сред и компаний, которые хотят централизованный сервер с гибкой настройкой. Однако Triton требует ручной настройки моделей и может уступать vLLM в скорости генерации текста, так как ориентирован на широкий класс задач (в т. ч. компьютерное зрение).

Общие выводы и рекомендации

- **Сравнение.**
- *vLLM* и *TGI* ориентированы на генерацию текста; vLLM обеспечивает максимальную пропускную способность и масштабируемость, а TGI — простоту развертывания и богатый набор дополнительных функций.

- *Ollama* и *llama.cpp* — это лёгкие решения для настольных компьютеров и небольших серверов. Ollama удобна и поддерживает reasoning-трассы и tool calling, в то время как `llama.cpp` позволяет запускать квантованные модели даже на CPU благодаря формату GGUF.
- *AutoAWQ* предоставляет удобный путь к 4-битному квантованию, позволяя разместить большие LLM на ограниченных GPU/CPU, но требует тщательной настройки и не подходит для больших батчей.

- *NVIDIA NIM* и *Triton* предлагают более широкий стек для корпоративных решений: NIM — полностью упакованные микросервисы с пред-обученными моделями, а Triton — гибкий сервер для любых ML-фреймворков.

- **Контейнеризация и планирование GPU.** Большинство решений поставляется в виде контейнеров (Docker), что облегчает развёртывание и переносимость. Для реальных кластеров необходимы системы планирования GPU (Kubernetes, Slurm, Ray) и мониторинг (Prometheus/Grafana). Важно предусмотреть динамический батчинг и распределение запросов, чтобы максимально загрузить GPU и снизить стоимость.

- **Выбор стека.** Набор инструментов зависит от задач и ресурсов. На небольшом сервере для экспериментов подойдут Ollama или llama.cpp с GGUF-моделями; для сервисов с тысячами запросов — vLLM/TGI; для корпоративной инфраструктуры — NIM и Triton. При этом *главное* — грамотно организовать процессы и данные: правильное описание задачи (конструкт RAG, формат ввода/вывода), оптимизация запросов, кеширование и батчинг. Именно качественные данные и workflow определяют качество результата, тогда как выбор конкретного движка влияет лишь на скорость и стоимость.

---

1 2 3 4 vLLM
https://docs.vllm.ai/en/latest/

5 6 7 8 Text Generation Inference
https://huggingface.co/docs/text-generation-inference/index

9 Quickstart - Ollama
https://docs.ollama.com/quickstart

10 Streaming - Ollama
https://docs.ollama.com/capabilities/streaming

11 Thinking - Ollama
https://docs.ollama.com/capabilities/thinking

12 Hardware support - Ollama
https://docs.ollama.com/gpu

13 14 15 raw.githubusercontent.com
https://raw.githubusercontent.com/ggml-org/llama.cpp/master/README.md

16 17 raw.githubusercontent.com
https://raw.githubusercontent.com/ggerganov/ggml/master/docs/gguf.md

18 19 20 21 raw.githubusercontent.com
https://raw.githubusercontent.com/casper-hansen/AutoAWQ/main/README.md

22 23 24 NVIDIA NIM Explained: The Fastest Way to Deploy AI Models in Production
https://greennode.ai/blog/greennode-nim-overview

25 26 27 28 raw.githubusercontent.com
https://raw.githubusercontent.com/triton-inference-server/server/main/README.md