ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ ПРОГРАММ ДЛЯ ЛОКАЛЬНОГО ЗАПУСКА БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ

Дичковский В. А., Ковалёв М. В.
Кафедра интеллектуальных информационных технологий,
Факультет информационных технологий и управления,
Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: {volodya.dichkovski, michail.kovalev7}@gmail.com

В данном докладе представлен сравнительный анализ трёх популярных локальных решений для работы с LLM – vLLM, Ollama и Llama.cpp. Для оценки производительности программ при генерации ответов использовались следующие метрики: время до первого токена, сквозная задержка , задержка между токенами, скорость генерации и количество запросов в секунду.

Введение

Современные большие языковые модели (LLM) получили широкое распространение, решая задачи от генерации текста до программирования и анализа данных.

Однако применение LLM через облачные API ограничено стоимостью, зависимостью от интернета, рисками для конфиденциальности и возможными утечками данных. Актуальным становится локальный запуск языковых моделей.

Производительность программ для локального запуска LLM в значительной степени определяется вычислительными ресурсами, в частности — мощностью видеокарты и объемом её видеопамяти. В профессиональных и промышленных решения, применяются высокопроизводительные и дорогостоящие графические ускорители, такие как NVIDIA A100 или H100, обладающие объемом видеопамяти соответственно 80, 94 гигабайт и выше. Эти устройства обеспечивают максимальную эффективность при работе с крупными моделями, но их крайне высокая стоимость делает их недоступными для большинства частных пользователей, в том числе студентов.

Поэтому остается актуальной задача выбора программы для локального запуска LLM, который обеспечит приемлемую производительность на персональных компьютерах и ноутбуках среднего класса, оснащённых более доступными видеокартами.

I. Выбор и анализ программ

Поскольку развертывание LLM часто происходит в системах реального времени, требующих низкой задержки ответов и высокой пропускной способности, важен анализ и выбор лучших программ[1].

Для анализа были выбраны три локальные программы – vLLM, Ollama и Llama.cpp – они активно применяются в практических задачах и имеют большую популярность.

vLLM демонстрирует высокую производительность за счёт эффективного пакетного объединения запросов (batch processing) и оптимизации использования видеопамяти.

Ollama отличается удобным интерфейсом и ориентирована на интеграцию в пользовательские приложения.

Llama.cpp представляет собой лёгкую реализацию, рассчитанную на работу с моделями при ограниченных вычислительных ресурсах, включая использование CPU, что делает её доступной для запуска на недорогих персональных компьютерах.

Все программы используют KV-кеш для повторного применения промежуточных значений слоёв трансформера, что снижает задержку и ускоряет генерацию. Также они поддерживают совместимый с OpenAI API интерфейс, ставший мировым стандартом.

II. Используемое оборудование

Тестирование проводилось с использованием видеокарты NVIDIA GeForce RTX 3060 (12 ГБ видеопамяти). Графические процессоры NVIDIA используют архитектуру с вычислительными ядрами CUDA. Это обеспечивает эффективное выполнение параллельных вычислений и широкую поддержку в современных фреймворках машинного обучения.

III. Выбор метрик для оценки

Современные LLM представляют собой нейросети, основанные на архитектуре трансформеров. Модель Transformer обрабатывает входные токены для генерации выходных прогнозов, поэтому метрики оценки её производительности связаны с характеристиками генерации токенов[2]. В статье [3] описаны наиболее важные метрики для оценки производительности вывода LLM. Для анализа были выбраны следующие метрики: Для оценки производительности моделей использовались ключевые метрики, описанные в [4 5]. Первая из них – Time to First Token (TTFT) – характеризует задержку между отправкой запроса и появлением первого токена. Этот параметр отражает скорость реакции системы и имеет реша-

ющее значение для приложений, чувствительных к задержка[6].

End-to-End Latency (E2E) отражает полное время отклика от начала запроса до завершения генерации текста. В отличие от TTFT, эта метрика охватывает весь цикл обработки данных и наиболее показательна для серверных нагрузок[7].

Inter-Token Latency (ITL) показывает средний интервал между последовательными токенами. Чем меньше это значение, тем плавнее воспринимается ответ модели в реальном времени.

Tokens Per Second (TPS) и Requests Per Second (RPS) характеризуют пропускную способность системы[8]. TPS показывает, насколько быстро модель генерирует токены, а RPS — сколько запросов она способна обслужить за единицу времени.

IV. Использование библиотеки Locust для нагрузочного тестирования

Для оценки производительности был выбран инструмент с открытым исходным кодом — Locust. Рассматривались также инструменты k6, Gatling, Artillery и NVIDIA GenAI-Perf.

Locust представляет собой библиотеку для Python, предназначенную для нагрузочного тестирования, которая позволяет моделировать поведение множества одновременно работающих пользователей. Библиотека полностью основана на событийной модели и использует лёгкие потоки через gevent для параллельного выполнения запросов.

Преимущества Locust включают веб-интерфейс, упрощающий настройку, возможность отслеживать результаты в реальном времени и экспорт отчетов в форматах html и csv.

V. Выбор модели для тестирования

Для тестирования выбрана модель Saiga LLaMA3 8B (8 миллиардов параметров) с квантованием q4_K. Она подходит для диалогов на русском языке и обеспечивает быстрые ответы при приемлемом качестве генерации. Использовался файл в формате GGUF.

Она была выбрана за лёгкость и способность обеспечивать быстрые ответы при сохранении приемлемого качества генерации. Для запуска использовался файл в формате GGUF.

VI. Результаты тестирования

Для каждой из программ тестирование проводилось в течение 5 минут, моделировалась нагрузка в 50 пользователей.

Перед каждым тестированием использовался warm up, который, согласно [2], повышает результаты производительности (до 4-5%).

Таблица 1 – Результаты тестирования производительности LLM

Метрика	vLLM	Ollama	Llama.cpp
Количество запросов	176	90	130
Среднее TTFT (мс)	30061.1	70700.2	52040.3
Средняе e2e latency (мс)	45444	73035.1	63251.6
Средне ITL (мс)	121.1	18.4	50.0
Среднее TPS (мс)	3.44	2.83	3.10
RPS	0.30	0.17	0.22

VII. Заключение

Проведенное тестирование показало, что vLLM лидирует по пропускной способности и общей задержке, оптимален для высоконагруженных API. Ollama уступает по скорости обработки, но обеспечивает минимальную задержку между токенами, что критично для плавности генерации и интерактивных приложений. Llama.cpp является сбалансированным универсальным решением. Метрики TPS и RPS наиболее определяют пропускную способность. Нужно стремиться сводить e2e latency к минимуму.

1. Best Practices for LLM Latency Benchmarking [Electronic resource] / Newline. – Mode of access: https://www.newline.co/@zaoyang/best-practices-for-llm-latency-benchmarking--257f132d. – Date of access: 16.10.2025.
2. Zhen, R. Taming the Titans: A Survey of Efficient LLM Inference Serving / R. Zhen [et al.] // arXiv preprint. – 2025. – arXiv:2504.19720v1.
3. Sagi, S. Optimizing LLM Inference: Metrics that Matter for Real Time Applications / S. Sagi // Journal of Artificial Intelligence & Cloud Computing. – 2025. – Vol. 4, № 4. – P. 1–10. – DOI: 10.47363/jaicc/2025(4)446.
4. Rajaneesh, N., Zollo, T., Zemel, R. Test-Time Warmup for Multimodal Large Language Models / N. Rajaneesh, T. Zollo, R. Zemel // arXiv preprint. – 2020. – arXiv:2509.10641v1. – P. 1–13.
5. LLM Benchmarking: Fundamental Concepts [Electronic resource] / NVIDIA. – Mode of access: https://developer.nvidia.com/blog/llm-benchmarking-fundamental-concepts/. – Date of access: 16.10.2025.
6. Sagi, S. Optimizing LLM Inference: Metrics that Matter for Real Time Applications / S. Sagi // Journal of Artificial Intelligence & Cloud Computing. – 2025. – Vol. 4, № 1. – P. 1–4.
7. NVIDIA NIM: Large Language Models Benchmarking [Electronic resource] / NVIDIA. – Mode of access: https://docs.nvidia.com/nim/large-language-models/1.2.0/benchmarking.html. – Date of access: 16.10.2025.
8. Тестирование производительности видеокарт на примере больших языковых моделей с использованием Llama.cpp [Электронный ресурс] / Habr. – Режим доступа: https://habr.com/ru/articles/916836/. – Дата доступа: 16.10.2025.