

Часть 5.5 — Аппаратная и платформенная дорожная карта (on-prem)

5.5.1 Модели развёртывания и инфраструктура

Кластер с NVIDIA GPU и NIM

- **NIM microservices.** NVIDIA предлагает набор микросервисов (NIM), предназначенных для развёртывания генеративных моделей на любой инфраструктуре с GPU. Это готовые, оптимизированные серверы, которые упрощают вывод и сокращают время до производства. NIM допускает запуск в облаке, центре обработки данных, локальной рабочей станции или на периферии и предлагает бесплатный доступ для разработки. Важно, что эти микросервисы включают продакшн-грейдовые рантаймы, регулярные обновления безопасности и поддерживают защищённую работу с данными¹². Таким образом, для on-prem-кластера NIM может служить базовой платформой для запуска модели и управления обновлениями и мониторингом.
- **Triton Inference Server.** Triton — открытый сервер выводов от NVIDIA, который обеспечивает единый слой для обслуживания моделей. Он поддерживает различные фреймворки (TensorFlow, PyTorch, ONNX, TensorRT), динамический батчинг и одновременное исполнение нескольких моделей. В нашем предложении Triton дополняет NIM: модели готовятся и оптимизируются в NIM, а Triton обеспечивает гибкое обслуживание и масштабирование внутри кластера.

Библиотеки для вывода LLM

- **vLLM.** vLLM — библиотека для быстрого вывода и сервинга LLM. Проект развит сообществом (изначально в Sky Computing Lab). vLLM обеспечивает *высокую пропускную способность* и использует PagedAttention для эффективного управления памятью KV-кэша; поддерживает *непрерывное батчирование* входящих запросов, *графы CUDA* для быстрого исполнения и имеет встроенную поддержку квантования (GPTQ, AWQ, INT4/INT8/FP8)³. Также vLLM поддерживает распределённый вывод, потоковую отдачу результатов и совместим с API OpenAI. Это делает vLLM хорошей опцией для локального сервера, где требуется высокая производительность на GPU или CPU.
- **Text Generation Inference (TGI).** TGI — сервер от Hugging Face, ориентированный на быстрый вывод больших языковых моделей. Он реализует динамическое батчирование, стриминг, offloading KV-кэша и умеет работать с несколькими GPU. В сочетании с vLLM TGI предоставляет гибкость: vLLM удобен для экспериментов и разработчиков, TGI — для production-сценариев с высокой нагрузкой и поддержкой HF ecosystem.

Квантование и оптимизация

- **AutoAWQ.** AWQ — метод низкоразрядного квантования весов. Документация vLLM предупреждает, что библиотека AutoAWQ устарела, поскольку её функциональность перенесена в проект `llm-compressor`, но всё ещё рекомендуется использовать AWQ для создания 4-битных моделей. Квантование переводит веса из формата BF16/FP16 в INT4, что резко снижает объём памяти; это даёт *меньшую задержку и использование памяти*³. Пользователь может установить AutoAWQ и самостоятельно квантовать модели, либо взять готовые AWQ-модели с Hugging Face³.
- **GGUF/llama.cpp.** Формат GGUF используется библиотекой `llama.cpp` для хранения квантизированных моделей и оптимизации вывода на CPU. `llama.cpp` поддерживает несколько режимов квантизации (4-, 5-, 8-бит), что позволяет запускать LLM на ноутбуках или серверах без мощных GPU. В on-прем-кластере CPU-нодам можно назначить сервисы генерации малых моделей (например, для предварительных фильтров), используя GGUF.

Векторные базы данных для RAG

- **Qdrant** — AI-нативная векторная база и поисковая система. Она позволяет извлекать смысл из неструктурированных данных, использовать семантический поиск и легко масштабируется. Qdrant open-source, можно развернуть локально или воспользоваться облачной версией; при этом она предоставляет удобный UI для взаимодействия с данными².
- **Milvus** — высокопроизводительная open-source векторная СУБД, ориентированная на приложения генеративного ИИ. Milvus обеспечивает быстрый поиск и масштабируется до десятков миллиардов векторов с минимальной потерей производительности⁴. Миллионеры используют её для RAG-систем, где требуется сверхбыстрый поиск контекстов.
- **pgvector** — расширение PostgreSQL, добавляющее операции со векторными данными (cosine/Euclidean расстояния) и индексы IVFFlat/HNSW. Хотя у нас нет явных цитат, pgvector позволяет хранить векторы рядом с реляционными данными и выполнять гибкие SQL-запросы. Это идеальный выбор для компаний, уже использующих Postgres.

Итоговая архитектура кластера

- **Аппаратная основа** — кластер из узлов с NVIDIA GPU (например, A100 или H100), под управлением Kubernetes или OpenShift. Использование GPU-серверов обеспечивает ускорение вывода и возможность обучения/дообучения небольших моделей.
- **Слой служебных сервисов.** Развёрнуты NIM microservices (или контейнеры с Triton/vLLM/TGI) для разных типов моделей: генерация кода, чат-ответы, извлечение эмбедингов, тестовая генерация. NIM позволяет быстро выпускать новые модели и автоматически получать обновления безопасности¹. Для стандартизации вывода используется vLLM или TGI: обе системы обеспечивают высокую производительность благодаря непрерывному батчингу, используют оптимизированные CUDA ядра и поддерживают различные варианты квантования³.

- **Слой данных.** Состоит из векторной БД (Qdrant/Milvus/pgvector) для хранения эмбеддингов кодовой базы и документации, плюс традиционной СУБД для метаданных. Это обеспечивает RAG-системам быстрый доступ к контексту.
- **Квантование и оптимизация.** Используются AWQ (или `llm-compressor`) для 4-битных моделей, GGUF для CPU-инференса, а также GPTQ/INT8 для компромисса между качеством и производительностью.
- **MLOps.** Для развертывания и обновления моделей — CI/CD-пайплайны (например, GitLab CI). Используются инструменты мониторинга (Prometheus + Grafana) и трассировки для контроля latency и ошибок.
- **Безопасность.** Все компоненты изолируются в Kubernetes; применяются политики сетевой безопасности, ограничивающие доступ к GPU-нодам. Для векторных баз задействуются средства шифрования и резервного копирования.

5.5.2 Предложение дорожной карты для Ростелекома

Чтобы внедрить AI-методологии в разработку и постепенно создать культуру использования LLM, рекомендуется следующий план:

1. **Пилотный проект.** Создать небольшую команду «Центр компетенций AI» и развернуть on-prem-кластер на базе 2–3 GPU-серверов. Использовать NIM/vLLM/TGI для обслуживания тестовых моделей, Qdrant или pgvector для хранения векторов. Начать с простой RAG-системы для поиска в кодовой базе.
2. **Интеграция в процессы разработки.** Разработчики подключают RAG-сервис к IDE для поиска и ответа на вопросы о легаси-коде. Используйте MCP (см. раздел 3) для безопасного доступа к кодовым репозиториям. Начать генерацию документации и тестов (см. раздел 5.2).
3. **Кадровый фактор.** Провести обучение команд: объяснить принципы LLM, RAG, безопасного промптинга и инженерии. Важно, чтобы тимлиды сами практиковали и демонстрировали применение ИИ; иначе разработчики не будут использовать инструменты.
4. **Управление данными.** Создать процесс регулярной индексации кодовых репозиторий, документов и баз знаний в векторную БД. Разработать политики очистки и обновления индекса.
5. **Расширение.** После успешного пилота:
 - Добавить новые модели (перевод, рефакторинг, генерация документации), опираясь на NIM и vLLM.
 - Внедрить AWQ/INT8 квантование для оптимизации ресурсов.
 - Расширить кластер, добавив дополнительные GPU-ноды и CPU-ноды для обслуживания легковесных задач.
 - Запустить процессы MLOps: автоматическое тестирование/каталогизация моделей, мониторинг качества, катастрофическое восстановление.
6. **Корпоративные стандарты.** Разработать внутренние политики по этике, безопасности и тестированию LLM (см. раздел 5.4). Применять фильтры, валидацию и меры против `prompt-injection`. Разработать стандарты документирования и с использованием LLM.

Заключение

Создание on-prem-кластера для генеративного ИИ в Ростелекоме требует сочетания аппаратной мощности (GPU-серверы), платформенных сервисов (NIM/Triton/vLLM/TGI) и грамотного выбора инструментов оптимизации (AWQ, GGUF). Векторные базы Qdrant или Milvus обеспечат быстрый доступ к корпоративным данным, а pgvector позволит интегрировать поиск с существующими системами. Предложенная дорожная карта призвана помочь стартовать с пилотного проекта, постепенно расширить инфраструктуру и сформировать AI-культуру среди разработчиков.

Источники

¹Источник: nvidia.com

²Источник: docs.nvidia.com

³Источник: docs.vllm.ai

⁴Источник: milvus.io