Рекомендации по инфраструктуре и производительности

Этот документ помогает выбрать конфигурацию для AI Assistant MVP (RAG: Qdrant + BM25 + LLM), от бюджетных до мощных. Указаны ориентиры по задержкам, требованиям к ОЗУ/CPU/диску и практикам оптимизации. Значения — эмпирические ориентиры для запросов с контекстом 4–8k токенов и генерацией ответа ~512 токенов.

Базовые допущения

- Поиск: Qdrant (COSINE, HNSW) + Whoosh (BM25F), фильтры по `space_id`/`doc_type`.
- LLM: локально (Ollama) или альтернативы (vLLM/TGI).
- Классификатор типов: правила MiniLM+логрегрессия; опционально zero-shot NLI.
- Узкое место по времени — генерация LLM (prefill + decode). Поиск обычно <1 с.

Конфигурации по уровням

A. Бюджетный GPU для 20B

- Цель: gpt-oss ~20B с приемлемой задержкой без больших затрат.
- GPU: 1× NVIDIA L4 24 GB или 1× A10G 24 GB.
- Ожидаемая задержка (512 токенов, Q4): 13–25 с (20–40 ток/с) + 1–2 с префилл.
- Ресурсы:
  - CPU: 8–16 vCPU.
  - RAM: 32–64 GB.
  - Диск (NVMe): 100–200 GB (LLM + HF/Ollama кэши + индексы).
- Подходит для: MVP/PoC, умеренная нагрузка, одиночные запросы/сек.

B. «Комфортный запас» для 20B/34B

- GPU: 1× A100 40 GB или 1× L40S 48 GB.
- Ожидаемая задержка (512 токенов, Q4):
  - 20B: 6–10 с + 1–2 с префилл.
  - 34B: 9–14 с + 2–3 с префилл.
- Ресурсы:
  - CPU: 8–16 vCPU.
  - RAM: 64 GB (комфортно под LLM + Qdrant + zero-shot).
  - Диск (NVMe): 200 GB.
- Подходит для: стабильный сервис, небольшая конкуррентность, хороший запас на рост.

C. 70B (в одно GPU)

- GPU: 1× A100 80 GB (предпочтительно одно большое GPU, чем 2×40 GB).
- Ожидаемая задержка (512 токенов, Q4/Q5): 10–20 с.
- Ресурсы: CPU 16 vCPU, RAM 64 GB, NVMe 200–400 GB.
- Примечание: с Ollama возможно, но vLLM/TGI дадут лучшее управление памятью и параллелизмом.

D. 120B и выше

- GPU: 2× H100 80 GB (минимум) или 4× A100 80 GB.
- Фреймворк: vLLM/TGI с тензор/пайплайн-параллелизмом (Ollama тут ограничен).
- Ожидаемая задержка (512 токенов): 12–25 с при хорошем квантизации и параллелизме.
- Ресурсы: CPU 16–32 vCPU, RAM 128 GB, NVMe 400+ GB.

Требования по памяти для Qdrant

Грубая оценка для 384-мерных эмбеддингов (float32), HNSW по умолчанию: - ~150–250 MB на каждые 100k векторов (вектора + граф + метаданные). - 100k чанков: ~0.2–0.5 GB; 500k: ~1–2.5 GB; 1M: ~2–5 GB. Реальный объём зависит от HNSW-параметров и payload.

Альтернативы Ollama: vLLM / TGI

- vLLM (OpenAI-совместимое API): быстрый префилл, Paged-KV, хорошая конкуррентность.
- TGI (HuggingFace Text Generation Inference): шардирование (`--num-shard`), оптимизации для больших моделей.
- Рекомендация: для 34B+ и особенно 70B/120B использовать vLLM/TGI.

Пример (vLLM) в docker-compose (GPU)

```
services:
  vllm:
    image: vllm/vllm-openai:latest
    ports: ["8001:8000"]
    deploy: {}
    runtime: nvidia
    environment:
      - NVIDIA_VISIBLE_DEVICES=all
      - NVIDIA_DRIVER_CAPABILITIES=compute,utility
    volumes:
      - ./.cache/hf:/root/.cache/huggingface
    command: >
      --model <hf-repo> --max-model-len 8192 --tensor-parallel-size 1
      --dtype float16 --gpu-memory-utilization 0.95
```

Backend меняем на `LLM_MODE=vllm` и `LLM_BASE_URL=http://vllm:8000/v1`.

Пример (TGI)

```
services:
  tgi:
    image: ghcr.io/huggingface/text-generation-inference:latest
    ports: ["8080:80"]
    runtime: nvidia
    environment:
      - NVIDIA_VISIBLE_DEVICES=all
      - NVIDIA_DRIVER_CAPABILITIES=compute,utility
    volumes:
      - ./.cache/hf:/data
    command: >
      --model-id <hf-repo> --dtype float16 --max-input-length 8192
      --max-total-tokens 9000
```

Backend настраиваем как `LLM_MODE=tgi`, `LLM_BASE_URL=http://tgi:80`.

Практики оптимизации

Контекст и генерация

- Сжимайте контекст: 3–6 лучших чанков по 300–600 токенов, избегайте лишних таблиц/списков.

- Используйте контекст-компрессию/summary-of-chunks для длинных документов.
- Снижайте `num_predict`, если достаточно кратких ответов.
- Держите модель «горячей» (keep-alive), квантизация Q4/Q5.

Поиск

- Тюнинг HNSW в Qdrant: `ef_search`, `ef_construct`, размер M — баланс точность/скорость.
- Храните индексы на NVMe, следите за IOPS.
- Для больших корпусов — добавляйте rerank (например, cross-encoder), но учитывайте его вклад во время.

Классификация (doc_type)

- Сначала правила/логрегрессия по MiniLM; zero-shot NLI вызывать только для `unknown`.
- Zero-shot на GPU: 10–150 мс на короткий текст (mDeBERTa-MNLI/BART-MNLI).

Параллелизм и масштабирование

- Для LLM используйте vLLM/TGI (очередь, тензор-параллелизм, шардинг).
- Backend (Uvicorn/Gunicorn): 2–4 воркера, по 1–2 worker-class/threads (по профилю нагрузки).
- Лимитируйте конкуррентность генераций, чтобы не уткнуться в VRAM.

Наблюдаемость и надёжность

- Таймауты: настройка `LLM_TIMEOUT` (по умолчанию 240 с), ретраи/бэкофы.
- Метрики: время prefill/decode, токен/сек, размер контекста, hit-rate ретрива.
- Кэширование: кэш ответов по (query, filters, top_k) на горизонте минут/часов.

Быстрые ответы на типовые вопросы

- Как сократить задержку в 2–3 раза без смены GPU?
  1) уменьшить `num_predict` до 256; 2) сократить контекст до 3–4 чанков; 3) держать модель прогретой; 4) Q4-квантизация; 5) перейти с Ollama на vLLM.
- Сколько RAM нужно кроме VRAM? Для уровней A/B: 64 GB — комфортно (LLM + Qdrant + zero-shot + кэши). Минимум 32 GB.
- Сколько диска? 100–200 GB для уровней A/B; 200–400 GB для 70B+.

Если нужна конкретная сборка под ваш облачный провайдер (AWS/GCP/Azure/RunPod/Lambda), дайте знать — подготовим `docker-compose.gpu.yml` и параметры модели под выбранный LLM-сервер (vLLM/TGI).