

Профилирование ASR-моделей на TEDLIUM: латентность, энергопотребление и узкие места

Корнилова Валерия Александровна

29 ноября 2025 г.

Постановка задачи

Цель: понять, как масштабируются ASR-модели по длине аудио и размеру батча, и какие части модели требуют оптимизации по времени и энергии.

Исследуем:

- Зависимость latency, RTF и энергопотребления от длины аудио и batch size.
- Компромисс скорость/качество (WER vs latency/RTF).
- Разницу поведения моделей на CPU и GPU (T4).
- Узкие места в пайплайне: препроцессинг, энкодер, декодер/CTC.

Модели ASR:

- `openai/whisper-tiny`
- `openai/whisper-base`
- `openai/whisper-small`
- `openai/whisper-medium`
- `openai/whisper-large-v3-turbo`
- `facebook/wav2vec2-base-960h` (CTC)

Датасет: TEDLIUM Release 3 (dev set)

- Англоязычные выступления TED, 16 кГц, моно.
- Используем dev-набор, учитываем флаг `ignore_time_segment_in_scoring`.

Оси масштабирования

Длина аудио (бакеты):

- 5 s: \approx 3–7 сек
- 15 s: \approx 10–20 сек
- 30 s: \approx 25–40 сек
- (опционально) 60 s: \approx 50–75 сек

Размер батча:

$$\text{batch_size} \in \{1, 4, 8, 16, 32\}$$

Для каждой комбинации (модель, устройство, длина, batch size) берётся фиксированное число примеров.

Методика эксперимента и метрики

Устройства:

- CPU: Colab без GPU
- GPU: Colab, T4 (cuda)

Метрики:

- Качество: WER, CER (jiwer).
- Скорость: latency per utterance, RTF.
- Энергия: энергия на utterance и на секунду аудио (мощность \times время).
- Вычислительная сложность: FLOPs (thop, fvcare).
- Память: max GPU memory.
- Breakdown по стадиям: препроцессинг, энкодер, декодер/СТС.

Качество распознавания (WER/CER)

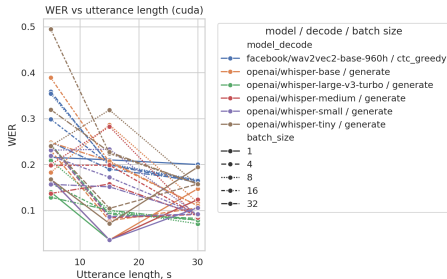
Средний WER по всем конфигурациям:

- `whisper-large-v3-turbo` → $\approx 11.9\%$ (лучшее качество)
- `whisper-small`, `whisper-medium` → $\approx 14.8\%$
- `whisper-base` → $\approx 18\%$
- `whisper-tiny` → $\approx 22\%$
- `wav2vec2-base-960h` → $\approx 22.6\%$

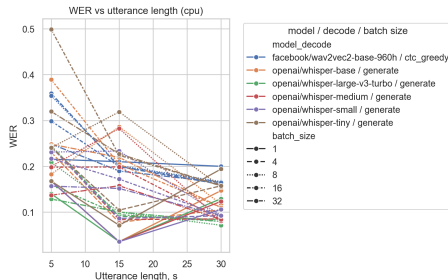
Вывод: качество монотонно растёт от `tiny` к `large-v3`, `whisper-small/medium` дают хороший баланс качества и размера.

WER vs длина аудио

GPU (T4)



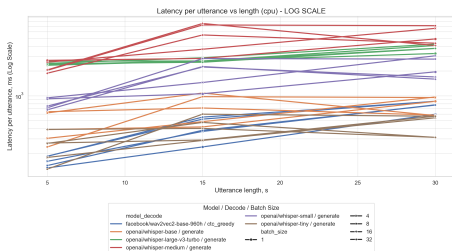
CPU



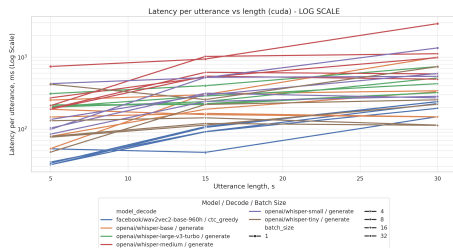
- WER растёт с длиной аудио для всех моделей.
- Порядок моделей по качеству одинаков на CPU и GPU.
- Устройство влияет на время и энергию, но не на WER.

Latency vs длина аудио (лог шкала)

CPU



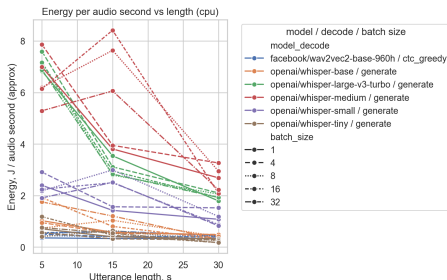
GPU (T4)



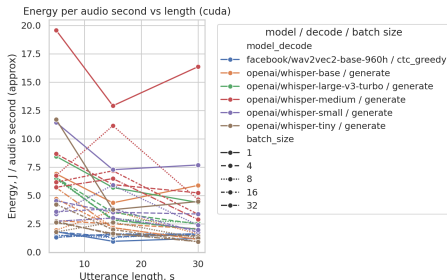
- На обеих платформах latency растёт почти линейно с длиной аудио.
- GPU даёт ускорение от $\sim 1.8\times$ (tiny) до $\sim 9\times$ (large-v3).
- Для длинных фрагментов крупные модели на CPU становятся слишком медленными.

Энергия vs длина аудио

CPU



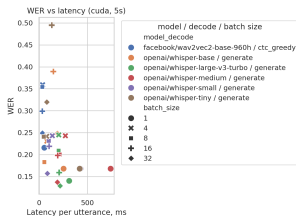
GPU (T4)



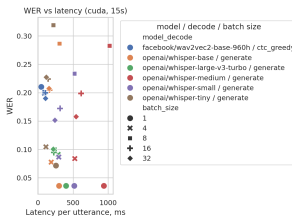
- Энергия растёт с длиной аудио для всех моделей.
- На GPU энергия на utterance выше, чем на CPU, особенно для лёгких моделей.
- Для тяжёлых моделей рост энергии на GPU умеренный относительно выигрыша по времени.

Trade-off: WER vs latency (GPU)

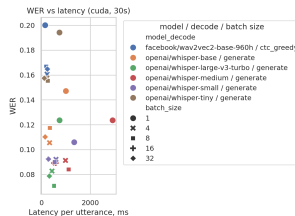
5 s



15 s

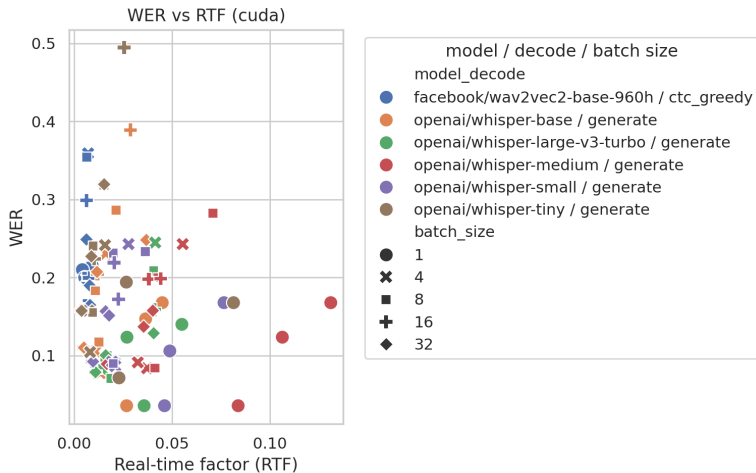


30 s



- Быстрые, но неточные: whisper-tiny, wav2vec2-base-960h.
- Медленные, но точные: whisper-large-v3-turbo, whisper-medium.
- whisper-small — разумный компромисс скорость/качество на GPU.

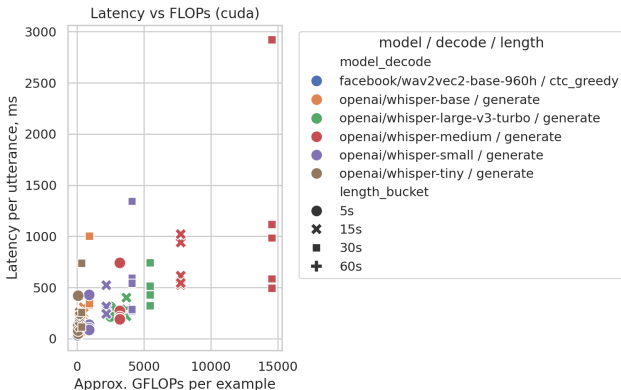
Trade-off: WER vs RTF (GPU)



- $RTF < 1$ — модель работает быстрее реального времени.

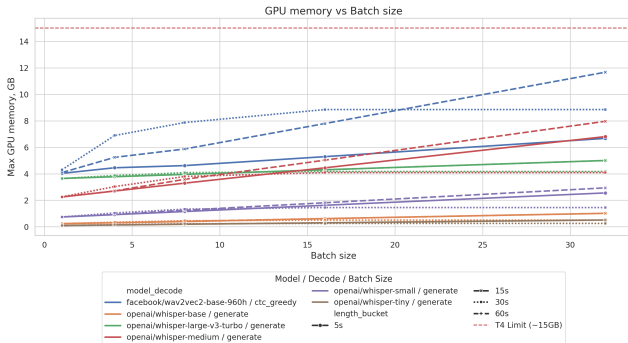
- На GPU большинство моделей имеют $RTF \ll 1$, особенно

Latency vs FLOPs (GPU)



- Чем больше FLOPs, тем в среднем выше latency.
- wav2vec2-base-960h — наименьшие FLOPs и одна из минимальных латентностей.
- У Whisper рост latency хорошо коррелирует с FLOPs: tiny → base

Использование GPU-памяти

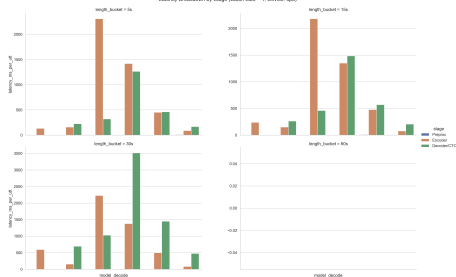


- Память растёт почти линейно с batch size.
- Крупные модели (medium, large-v3) быстрее упрутся в лимит памяти T4.
- Для практики: умеренные batch size (4–16) для крупных моделей.

Разбиение latency по стадиям

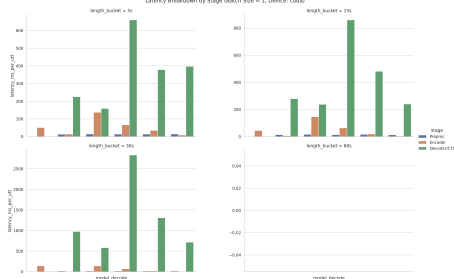
CPU

Latency Breakdown by Stage (Batch Size = 1, Device: cpu)



GPU (T4)

Latency Breakdown by Stage (Batch Size = 1, Device: cuda)



- **wav2vec2-base-960h**: узкое место — энкодер (до 99% времени).
- **Whisper на CPU**: значимый вклад и энкодера, и декодера (особенно для длинных аудио).
- **Whisper на GPU**: основное время уходит в авторегрессионный декодер.

Практические рекомендации

Онлайн, короткие команды, ограниченные ресурсы:

- `whisper-tiny`, `whisper-base`, `wav2vec2-base-960h` на CPU или слабом GPU.
- Минимальная latency, невысокие требования к памяти, WER 18–23%.

Стримы, конференции (баланс скорость/качество, есть GPU):

- `whisper-small` как модель по умолчанию.
- `whisper-base` — если качество менее критично, но важна минимальная задержка.

Оффлайн транскрибация с упором на качество (лекции, подкасты):

- `whisper-large-v3-turbo`, `whisper-medium` на GPU.
- Можно использовать умеренные batch size, запускать задачи оффлайн

Оптимизация времени и энергии

Основные узкие места:

- CTC-модель (wav2vec2): энкодер.
- Whisper: авторегрессионный декодер (особенно на GPU и для длинных аудио).

Возможные направления оптимизации:

- Квантование, pruning, distillation энкодера.
- Упрощение декодера Whisper: уменьшение beam size, max_new_tokens, ранняя остановка.
- Умеренный batching на GPU для увеличения throughput и энергоэффективности.

- Профилированы 6 ASR-моделей на TEDLIUM dev на CPU и GPU T4.
- Показано, как масштабируются латентность, RTF, энергия и память по длине аудио и batch size.
- GPU ускоряет inference до $9\times$, особенно у крупных моделей Whisper.
- CPU энергоэффективнее по энергии; GPU даёт многократный выигрыш по времени, особенно для тяжёлых моделей Whisper.
- Узкие места: энкодер у CTC и декодер у Whisper на GPU.

Спасибо за внимание!