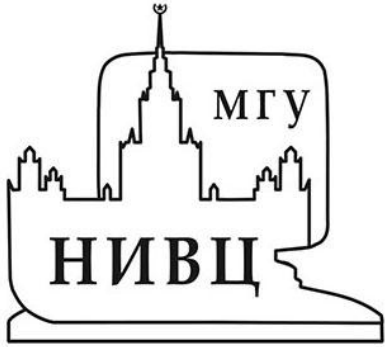


Обучение языковых моделей в multi-node / multi-gpu конфигурации

Тихомиров М.М.
Чернышев Д.И.

НИВЦ МГУ имени М. В. Ломоносова

Large Language Models

Большие языковые модели (LLM) представляют собой нейросеть на основе архитектуры трансформер.

- Современные размеры: 7, 13, 30, 70 ... **миллиардов параметров**.
- Обучалась предсказывать следующее слово на **триллионах слов**.

Регулярно появляются новые все более качественный LLM (мультиязычные / англоязычные). Основа роста качества LLM - наращивание вычислительных мощностей.

В Opensource появляются модели вплоть до **405 миллиардов** параметров, которые обучались на **15 триллионах** (LLaMa-3.1-405B). Только чтобы запустить ее с квантизацией (так как в “полном” виде не влезет) необходим узел с 8 A100/H100.

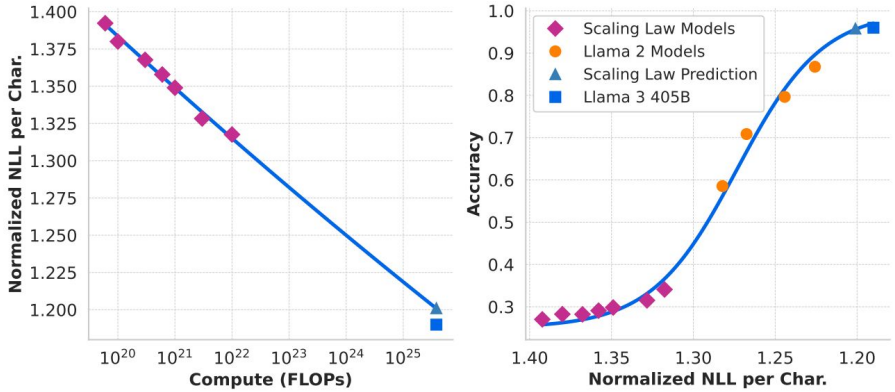
Сколько стоит GPT-3

Model	Total train compute (PF-days)	Total train compute (flops)	Params (M)	Training tokens (billions)	Flops per param per token	Mult for bwd pass
BERT-Base	1.89E+00	1.64E+20	109	250	6	3
BERT-Large	6.16E+00	5.33E+20	355	250	6	3
RoBERTa-Base	1.74E+01	1.50E+21	125	2,000	6	3
RoBERTa-Large	4.93E+01	4.26E+21	355	2,000	6	3
GPT-3 Small	2.60E+00	2.25E+20	125	300	6	3
GPT-3 Medium	7.42E+00	6.41E+20	356	300	6	3
GPT-3 Large	1.58E+01	1.37E+21	760	300	6	3
GPT-3 XL	2.75E+01	2.38E+21	1,320	300	6	3
GPT-3 2.7B	5.52E+01	4.77E+21	2,650	300	6	3
GPT-3 6.7B	1.39E+02	1.20E+22	6,660	300	6	3
GPT-3 13B	2.68E+02	2.31E+22	12,850	300	6	3
GPT-3 175B	3.64E+03	3.14E+23	174,600	300	6	3

Для обучения GPT-3 175B (**3640 PF-days,** $4.6M-$12M) потребовалось бы **7 месяцев** обучения на **512 V100**, или **43 дня** на **512 A100** (₽70M и 112 месяцев на Volta-1).

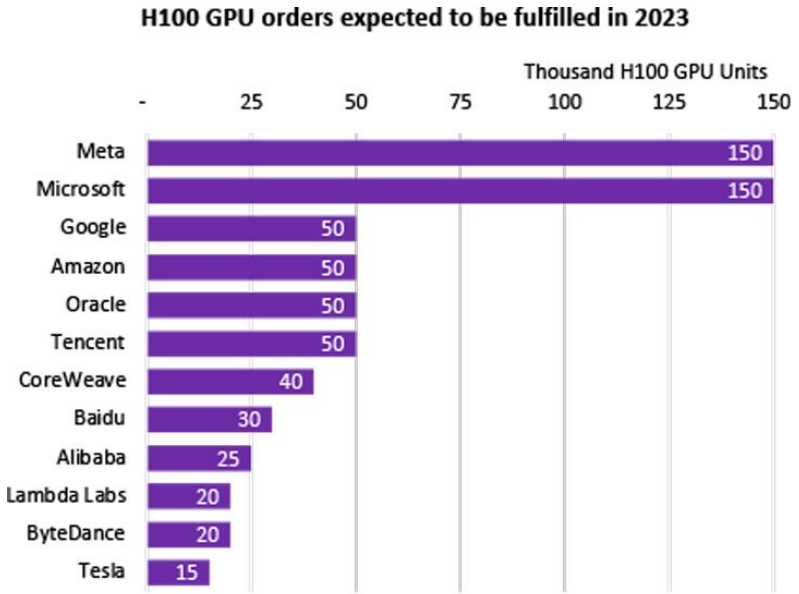
Стоимость обучения InstructGPT: **4.9 PF-days** для **SFT** и **60 PF-days** для **PPO-ptx**.

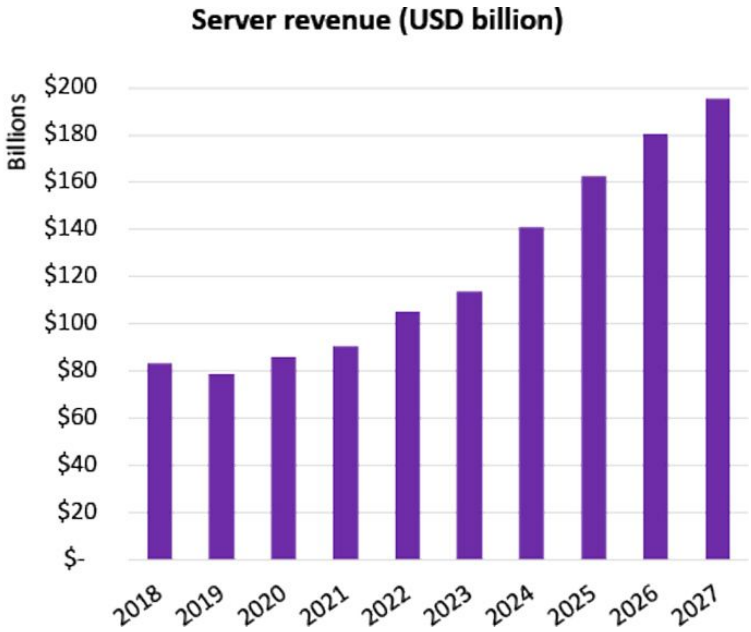
Сколько стоит LLaMa-3.1-405B



- Обучение модели стоило **3.8 × 10^25** FLOPs или **38 иоттафлопс.**
- Использовался кластер из **16000 H100**
- В 100 раз “дороже”, чем GPT-3 175B

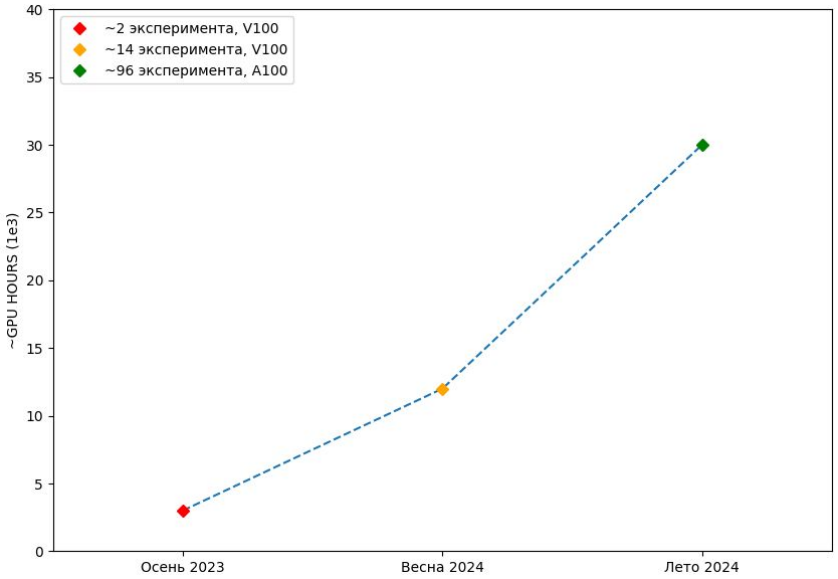
Закупка GPU в мире





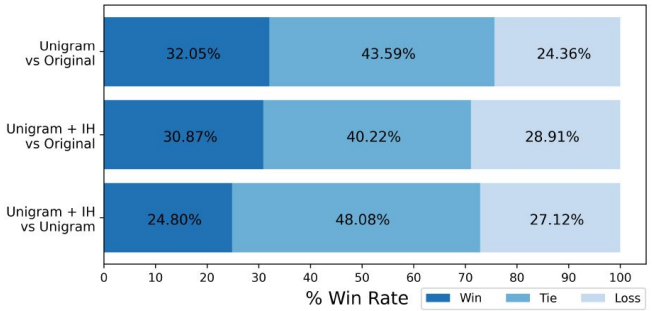
Наш опыт обучения LLM

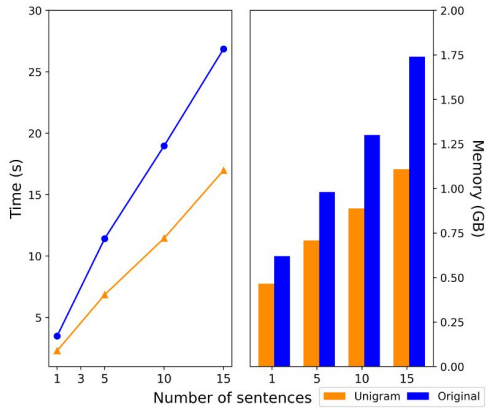
- Работа преимущественно с “небольшими” моделями в 7 миллиардов параметров.
- Задача не обучить с нуля, а адаптировать уже существующую модель на русский язык.
- Три итерации экспериментов с трендом на увеличение GPU часов.



Impact of Tokenization on LLaMa Russian Adaptation

- Работа осенью 2023 года
- Эксперименты проводились на Ломоносов-2 DGX-2 (16xV100)
- В условиях ограниченных ресурсов выбор гиперпараметров был “интуитивным”
- Были получены первые модели серии ruadapt на базе LLaMa-2

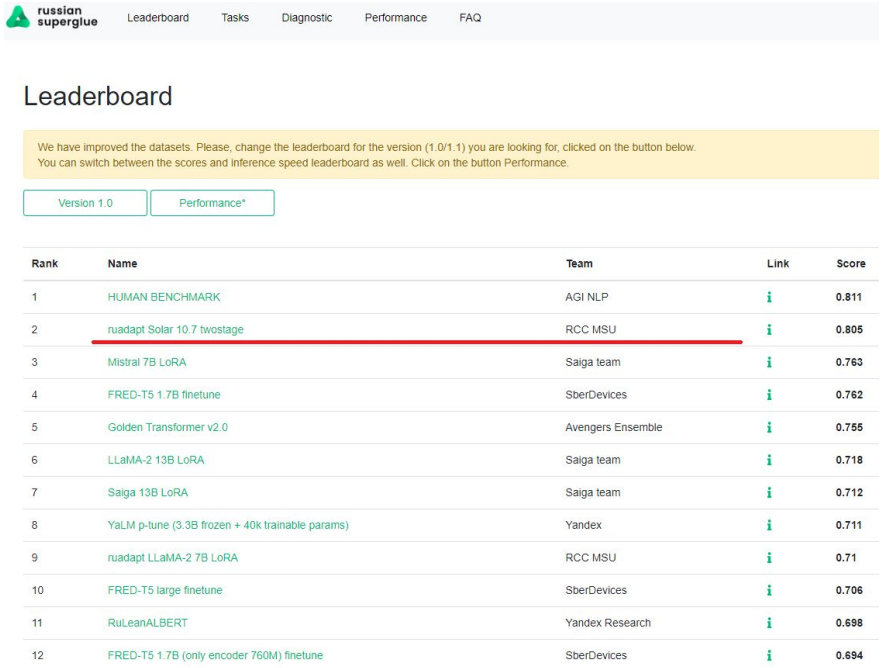




Tikhomirov M., Chernyshev D. Impact of Tokenization on LLaMa Russian Adaptation //2023 Ivannikov Ispras Open Conference (ISPRAS). – IEEE, 2023. – С. 163-168.

Improving Large Language Model Russian Adaptation with Preliminary Vocabulary Optimization

- Работа зимой/весной 2024 года, модель на 10.7 млрд. параметров
- Эксперименты проводились на Ломоносов-2 Volta1 (10x2xV100) и узле МГУ270 (8xA100)
- В случае Volta1 был отработан **multi-node** подход. Пропускная способность была в 2 раза ниже, чем в случае DGX-2 при одинаковом количестве GPU.



Rank	Name	Team	Link	Score
1	HUMAN BENCHMARK	AGI NLP		0.811
2	ruadapt Solar 10.7 twostage	RCC MSU		0.805
3	Mistral 7B LoRA	Saiga team		0.763
4	FRED-T5 1.7B finetune	SberDevices		0.762
5	Golden Transformer v2.0	Avengers Ensemble		0.755
6	LLaMA-2 13B LoRA	Saiga team		0.718
7	Saiga 13B LoRA	Saiga team		0.712
8	YaLM p-tune (3.3B frozen + 40k trainable params)	Yandex		0.711
9	ruadapt LLaMA-2 7B LoRA	RCC MSU		0.71
10	FRED-T5 large finetune	SberDevices		0.706
11	RuLeanALBERT	Yandex Research		0.698
12	FRED-T5 1.7B (only encoder 760M) finetune	SberDevices		0.694

Tikhomirov, M.M. and Chernyshev, D.I., 2024. Improving Large Language Model Russian Adaptation with Preliminary Vocabulary Optimization. Lobachevskii Journal of Mathematics

Facilitating large language model Russian adaptation with Learned Embedding Propagation

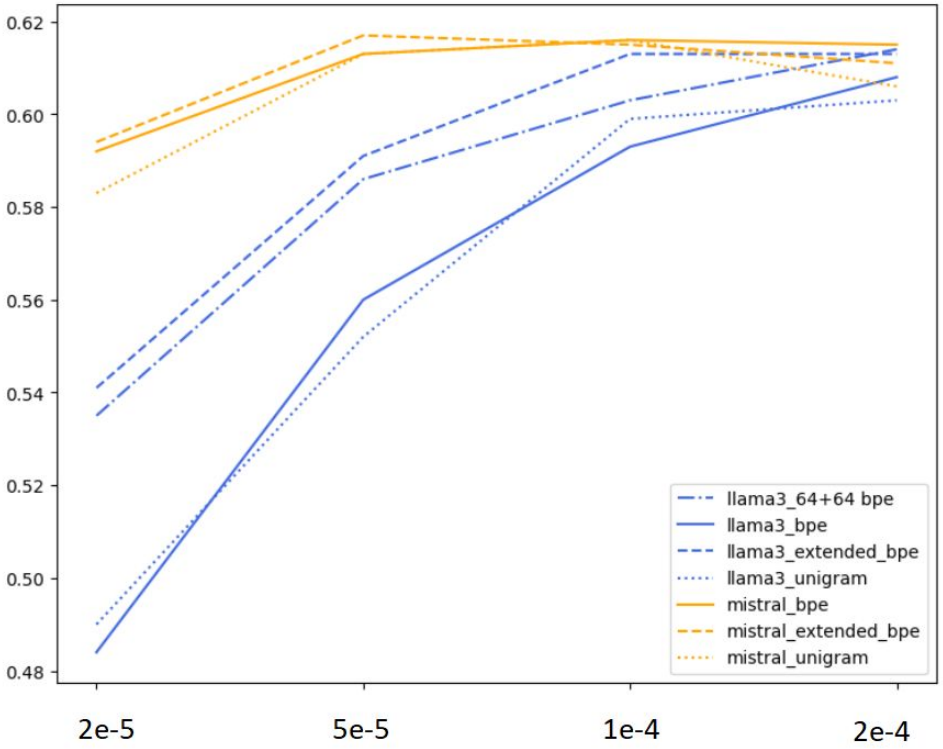
- Был получен доступ к кластеру МГУ-270
- Эксперименты запускались на 8 узлах по 8 A100 на каждом (64 A100)
  - Подход к распараллеливанию обучения: DDP (Distributed Data Parallel)
- Подробно исследовали:
  - Влияние способа адаптации токенизации: BPE, Unigram, расширение исходного словаря
  - Гиперпараметры запуска (learning rate, batch size)
  - Две наилучшие на момент экспериментов модели: LLaMa-3 и Mistral-7B-v0.1
- Предложили новый метод **LEP (Learned Embedding Propagation)** для адаптации инструктивных моделей

Tikhomirov, M.M. and Chernyshev, D.I., 2024. Facilitating large language model Russian adaptation with Learned Embedding Propagation

Learned Embedding Propagation: исследование гиперпараметров

- Исследовали зависимость качества от гиперпараметра **скорости обучения** и алгоритма токенизации

- Эксперименты показали, что чем **более обучена модель**, тем **тяжелее идет адаптация**: качество для llama-3 существенно зависит от скорости обучения



Learned Embedding Propagation: основная идея

- Языковые модели условно можно разделить на **базовые (foundation)** и **инструктивные**. Последние получаются из базовых путем специальной процедуры **instruction-tuning**, которая оказывает существенное влияние на качество моделей.

- **Проблема** предыдущих подходов к адаптации: мы **теряем** успешные **инструктивные версии** и нужно обучать их заново, что не всегда возможно.

- Идея LEP: адаптировать только связанные с токенизацией **слои-эмбеддинги**, а затем их **проецировать** на соответствующие инструктивные модели

Learned Embedding Propagation

- Были проведены эксперименты с парами моделей:
  - LLaMa-3-8B и LLaMa-3-8B-instruct
  - Mistral-7B-v0.1 и openchat-3.5-0106

- Был создан новый бенчмарк для быстрой оценки LLM и написан фреймворк для оценки (llmtf_open)

- Результаты показали, что предложенный метод позволяет достичь исходного качества моделей и даже превзойти его, при этом требуя лишь небольшого этапа дообучения на инструктивных данный для “калибровки”

https://github.com/RefalMachine/llmtf_open

Learned Embedding Propagation

Darumeru few-shot evaluation results for best language-adaptation checkpoints.

Model	Vocab	Symbols per token	Micro-Avg	DaruMMLU	DaruMERA	DaruSum	DaruCopy (EN)	DaruCopy (RU)
Mistral-7B	**original**	2,44	0,604	**0,545**	0,504	0,307	**1,000**	**1,000**
	BPE	**3,76**	0,616	0,528	0,537	**0,316**	0,995	0,984
	Unigram	**3,78**	0,614	0,516	**0,544**	0,311	0,995	0,960
	Extended	**3,77**	**0,617**	**0,538**	0,532	0,314	**1,000**	**0,995**
LLaMa-3-8B	**original**	2,89	**0,629**	**0,582**	0,547	0,326	0,980	0,982
	BPE	**4,4**	0,618	**0,561**	0,532	0,321	1,000	0,963
	Unigram	**4,35**	0,609	0,560	0,517	0,316	1,000	0,951
	Extended	3,78	**0,627**	0,560	0,550	0,325	0,980	0,983
	Optimized	3,4	**0,620**	0,552	0,536	0,323	0,981	0,989

Benchmark results for continued instruction-tuning of LEP-Conversion models

Model	dataset	Micro-Avg	DaruMMLU	DaruMERA	DaruSum	DaruCopy (EN)	DaruCopy (RU)
OpenChat 3.5 (original)	-	0,607	**0,543**	0,526	0,322	0,999	0,917
	saiga d7	0,611	0,540	**0,528**	**0,325**	0,999	0,945
	+copy task	**0,615**	0,541	0,524	0,324	**1,000**	**0,995**
OpenChat 3.5 (Unigram)	-	0,564	0,515	0,519	0,301	0,998	0,646
	saiga d7	0,599	**0,532**	0,556	0,316	0,999	0,754
	+copy task	**0,630**	0,530	**0,559**	**0,321**	**1,000**	**0,999**
OpenChat 3.5 (Extended)	-	0,610	0,535	0,541	0,307	0,999	0,914
	saiga d7	0,616	**0,543**	**0,566**	0,319	0,999	0,845
	+copy task	**0,632**	0,541	0,563	**0,321**	**1,000**	**0,989**
LLaMa-3-8B instruct (original)	-	0,61	0,571	0,510	0,322	1,000	0,972
	saiga d7	0,615	**0,576**	0,512	0,329	1,000	0,983
	+copy task	**0,616**	0,575	**0,513**	**0,332**	**1,000**	**0,995**
LLaMa-3-8B instruct (Extended)	-	0,603	0,557	0,503	0,328	0,990	0,956
	saiga d7	0,614	**0,568**	0,519	0,338	0,995	0,961
	+copy task	**0,618**	0,565	**0,521**	**0,339**	**1,000**	**0,984**
LLaMa-3-8B instruct (Optimized)	-	0,603	0,553	0,506	0,326	0,995	0,949
	saiga d7	0,611	**0,555**	0,515	0,336	**1,000**	0,971
	+copy task	**0,617**	**0,555**	**0,522**	**0,339**	**1,000**	**0,989**

Докер контейнер

- Pytorch как основной фреймворк для обучения нейросетей.
- Пакеты transformers, accelerate, peft для работы с языковыми моделями.
- Собирал на основе контейнера **nvcr.io/nvidia/pytorch:24.04-py3**
  - Когда пробовал собирать на основе другого контейнера, не от NGC, были проблемы при multi-node запусках, скорость вычислений существенно падала.
  - Все python пакеты и утилиты, которые были нужны, ставил поверх через Dockerfile.

- **Ubuntu 22.04** including **Python 3.10**
- **NVIDIA CUDA 12.4**
- **NVIDIA cuBLAS 12.4.5.8**
- **NVIDIA cuDNN 9.1.0.70**
- **NVIDIA NCCL 2.21.5**
- NVIDIA RAPIDS™ 24.02
- **rdma-core 39.0**
- NVIDIA HPC-X 2.18
- **OpenMPI 4.1.4+**
- GDRCopy 2.3
- TensorBoard 2.9.0
- **Nsight Compute 2024.1.0.13**
- **Nsight Systems 2024.2.1.38**
- **NVIDIA TensorRT™ 8.6.3**
- **Torch-TensorRT 2.3.0a0**
- **NVIDIA DALI® 1.36**
- **nvImageCodec 0.2.0.7**
- **MAGMA 2.6.2**
- **JupyterLab 2.3.2** including **Jupyter-TensorBoard**
- **TransformerEngine** 1.5
- PyTorch quantization wheel 2.1.2

https://docs.nvidia.com/deeplearning/frameworks/pytorch-release-notes/rel-24-04.html

Dockefile

```
FROM nvcr.io/nvidia/pytorch:24.04-py3
RUN pip install transformers==4.37.2
RUN pip install peft==0.9.0
RUN pip install jupyterlab
RUN pip install packaging
RUN pip install ninja
RUN pip install datasets==2.18.0
RUN apt-get update && apt-get upgrade -y && apt-get install -y git
RUN apt-get install nano
RUN pip install flash-attn==2.5.0 --no-build-isolation
RUN pip install accelerate==0.26.0
RUN pip install numpy
RUN pip install pandas
RUN pip install tqdm
RUN pip install evaluate
RUN pip install deepspeed
RUN pip install tensorboard
RUN pip install scikit-learn
RUN pip install sentencepiece

WORKDIR /workdir
```

sbatch

Запускаю через команду sbatch.

```
#!/bin/bash

#SBATCH --nodes=8                   # number of nodes
#SBATCH --gres=gpu:8                # number of GPUs per node
#SBATCH --time=4-23:59:59

export NNODES=8
export GPUS_PER_NODE=8

export head_node=( $( scontrol show hostnames $SLURM_JOB_NODELIST ) )
export head_node_ip=$(srun --nodes=1 --ntasks=1 --gpus 0 -w "$head_node" hostname --ip-address)

export HF_HOME=/scratch/tikhomirov/workdir/data/.cache/

echo Head Node: $head_node
echo Head Node IP: $head_node_ip
echo "${head_node_ip: -1}"
srun --container-image /scratch/tikhomirov/ngc_cuda_pytorch_24_04_v1+latest.sqsh --container-workdir /scratch/tikhomirov --container-mounts /scratch/tikhomirov:/scratch/tikhomirov bash -c "cd /scratch/tikhomirov/workdir/projects/ruadapt_training  && ./run_train_peft_32_128.sh"
```

- Конфигурация узлов и GPU по-умолчанию, а также лимит времени задается через #SBATCH
- head_node_ip нужен для синхронизации в скрипте запуска обучения через torchrun
- Логи пишутся в slurm-task_id.out

Заключение

- Развитие ИИ тесно связано с вычислительными возможностями

- Только для того, чтобы проводить эксперименты с моделями размером 7-10 миллиардов параметров требуется кластер оснащенный современными вычислителями: десятки A100 / H100

- На текущий момент развитие LLM в России почти полностью связано с использованием зарубежных наработок, обучать модели подобного качества с нуля не представляется возможным, так как требуется:
  - Тысячи GPU объединенных в единый кластер
  - Специалисты, которые умеют обучать подобные модели