Széchenyi Plan Plus Let's build Hungary together!

Government of Hungary

Funded by the European Union NextGeneration EU

ARTIFICIAL INTELLIGENCE National Laboratory

Transzformer modellek tanításának hatékonyságának növelésével kapcsolatos megoldások és tapasztalatok

Ficsor Tamás

SZTE UNIVERSITY OF SZEGED

Területem

- NLP (első sorban transzformer) modellek
  - értelmezhetősége
  - hatékony inferenciája és
  - tanítása
- HPC (High Performance Computing) tudás Fugaku használatán keresztül

Gyakran használt környezet

- PyTorch
- Accelerate
- PyTorch-Lightning
- Transformers
- Wandb

1. Számunkra hatékony implementációk megválasztása
2. Modellek monitorozása
3. Párhuzamosítás
4. Egyéb lehetőségek

Számunkra hatékony implementációk megválasztása

Implementációs tényezők

- Transformers könyvtár által használt implementációk nem mindig a legoptimálisabbak
  - Például: Bert, RoBERTa, GPT2…
    - FlashAttention, Unsloth, llama.cpp, triton, cuda
- Tanítást folyamatot elrejtő implementációk sokszor lassabbak
  - Például: **transformers.Trainer** vagy **pytorch-lightning.Trainer**
  - Túl sok felesleges és elrejtett viselkedés
- Operációegyesítés (Fused Kernels)
  - **torch.compile (torch >= 2.0)**

Bit reprezentációk

- Precízió (FP32, FP16, BF16, TF32)
  - Ha lehetséges akkor válasszuk a BF16-ot az FP16 helyett
  - **torch.amp** (Automatic Mixed Precision)**, accelerate, deepspeed, bitsandbytes**

Format	Exponent	Mantissa	Mantissa^2	O(Transistors)	
float32	8	23	529	537	
float16	5	10	100	105	5x
bfloat16	8	7	49	57	10x
Float8 E4M3	4	3	9	13	40x
Float8 E5M2	5	2	4	9	60x
MXFP4 E2M1	2	1	1	3	180x

Bit reprezentációk

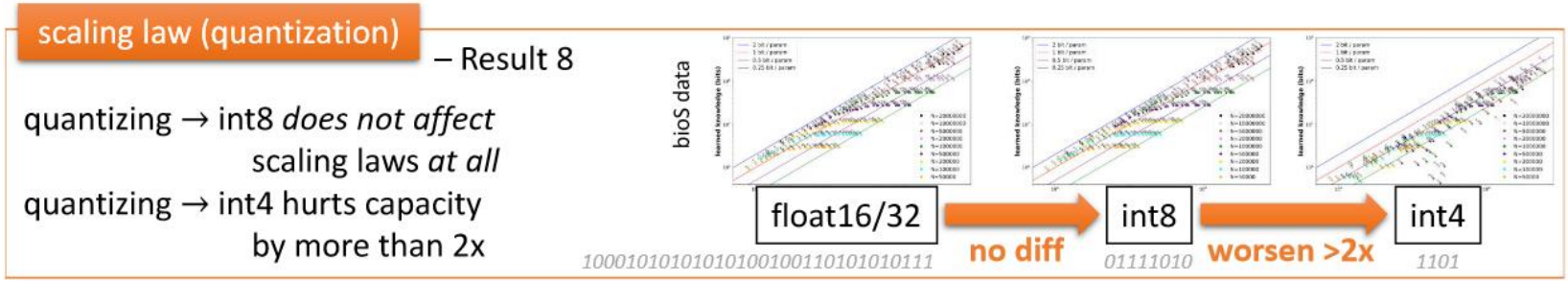


Figure 6: Illustration of Result 8 (see Figure 14 for bioS data and Figure 15 for bioD data in Appendix C).

Allen-Zhu, Zeyuan, and Yuanzhi Li. "Physics of language models: Part 3.3, knowledge capacity scaling laws." *arXiv preprint arXiv:2404.05405* (2024).

Memóriát befolyásoló tényezők

- Optimalizáló megválasztása
  - **AdamW**, jobb konvergencia, nagyobb memória lenyomat
  - **Adafactor**, rosszabb konvergencia, szignifikánsan kisebb memórialenyomat
  - 8-bit optimalizálók (8-bit AdamW)
- Gradiens akkumuláció
- Gradient checkpointing[1]

[1] Fitting larger networks into memory. Yaroslav Bulatov https://medium.com/tensorflow/fitting-larger-networks-into-memory-583e3c758ff9

Egyéb tanácsok

- Batch-eket igyekezzünk dinamikusan egységnyi hosszúra hozni, ha lehetséges
- Kerüljük (amennyire csak lehet) a szinkronizációs parancsokat
  - **.print, .cuda(), .cpu(), .item(), vagy python control flow**
- Tensor-t csak a cél eszközön hozzunk létre
- Memória allokátor: tcmalloc (vagy jemalloc)

Monitorozás

Profilozás és Monitorozás

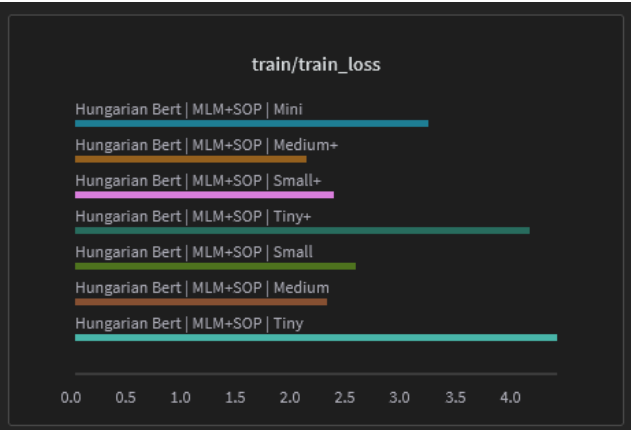
- **torch.monitor** és **torch.profiler**
- Nvidia-smi (vagy **nvidia-ml-py3**)
- Python: **profile** és **cProfile** (**flameprof** vizualizáláshoz)
- Perf (ha nagyon bátor az ember)

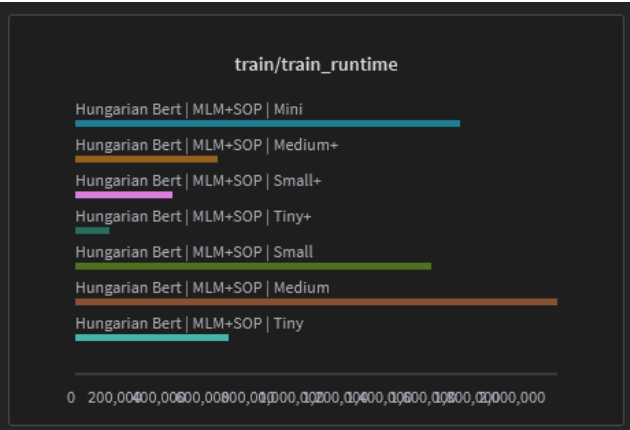
Teljesítmény és logolás

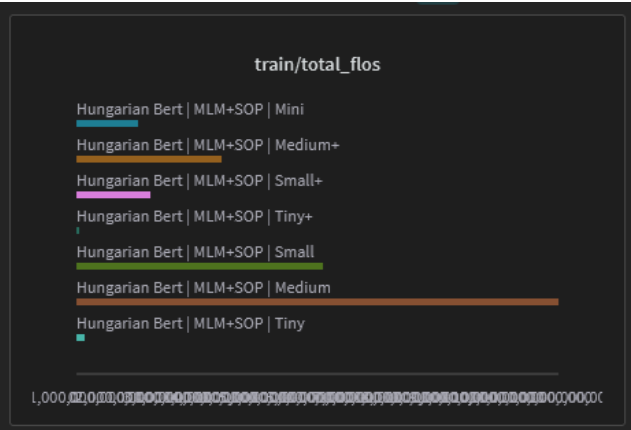
- Wandb (Weights&Biases)
  - Rendszer és Model használati statisztikák logolása
  - Model teljesítményének és egyéb statisztikáinak nyomon követése
  - Valós idejű online keretrendszer
- TensorBoard
- Aim[1]

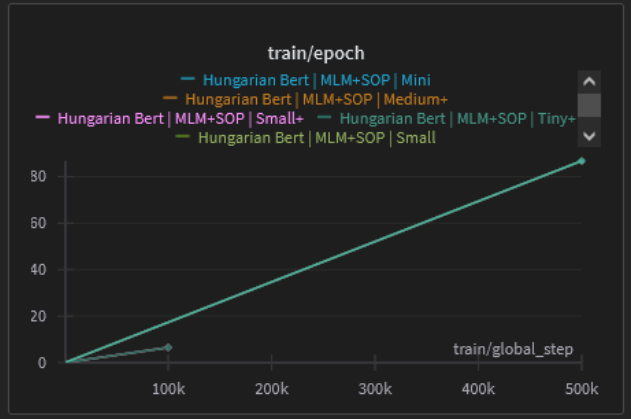
[1] https://github.com/aimhubio/aim

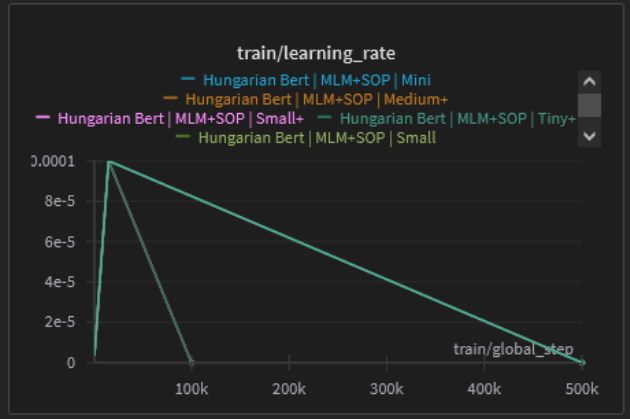
Wandb - Model teljesítmény

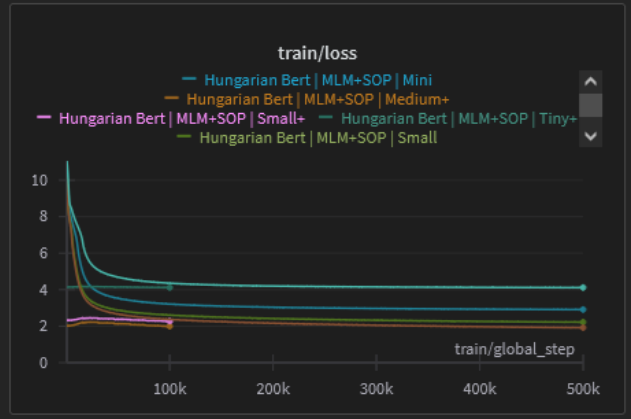






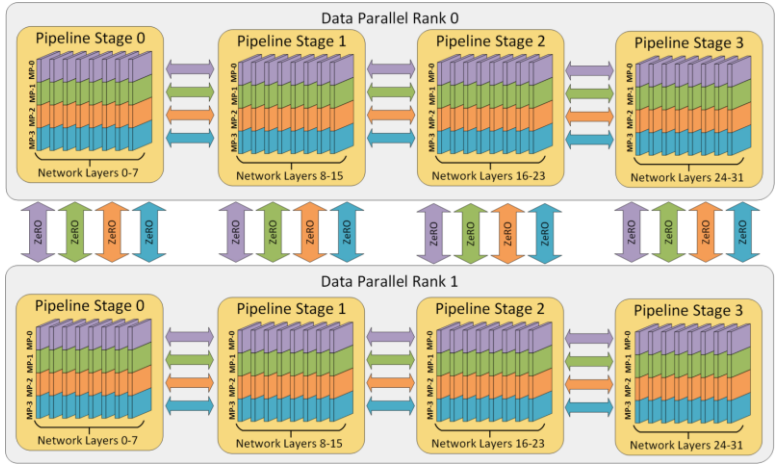






Párhuzamosság

Tipusok

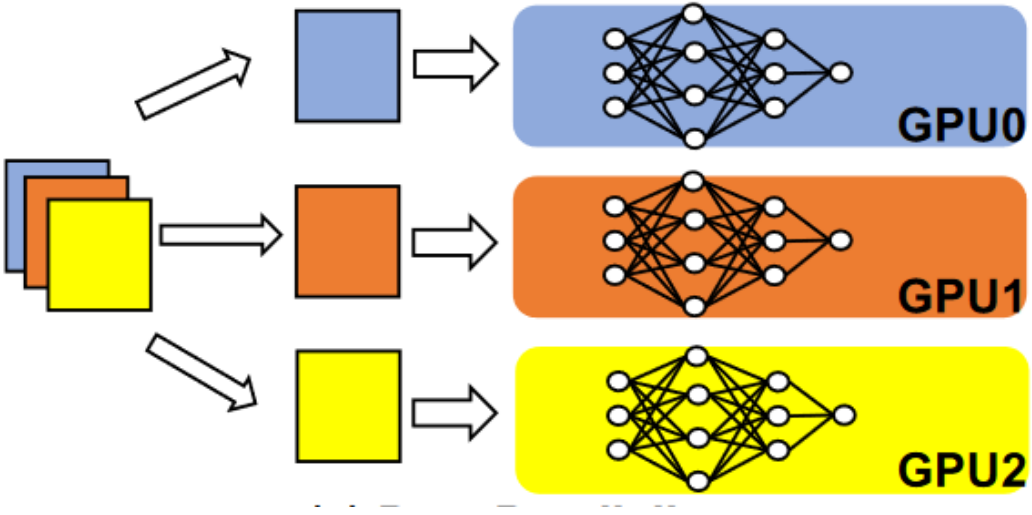


- Adat
- Pipeline
- Tensor
- Optimizer

Adatpárhuzamosság

Megvalósítás:

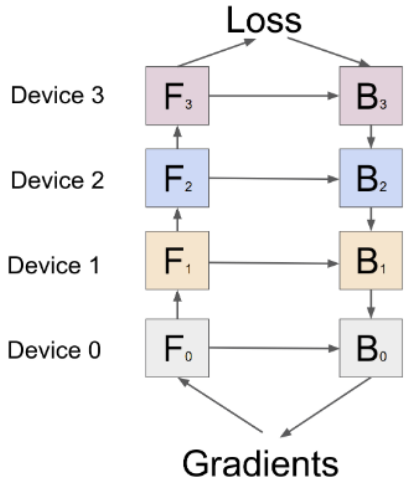
- PyTorch: **DataParallel** és **DistributedDataParallel** osztályokkal
- **accelerate**, **deepspeed**, **pytorch-lightning**

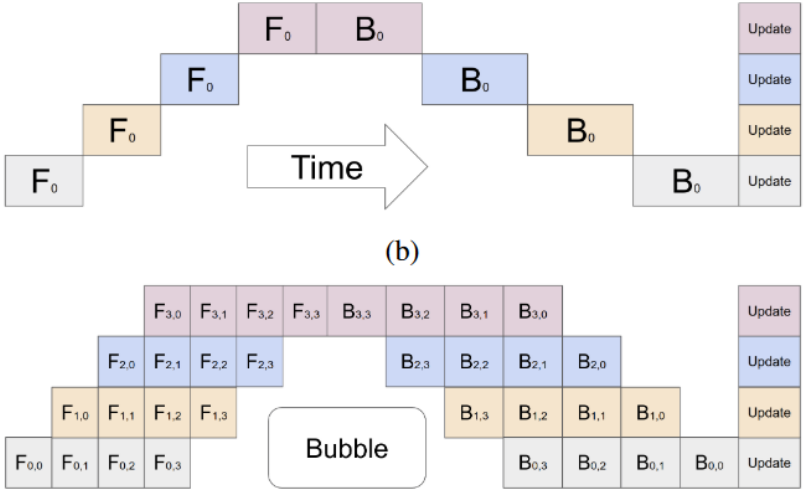


Pipeline párhuzamosság

Megvalósítás:

- PyTorch: **torch.distributed.pipelining** (alpha)
- **deepspeed, nanotron**, **megatron,**

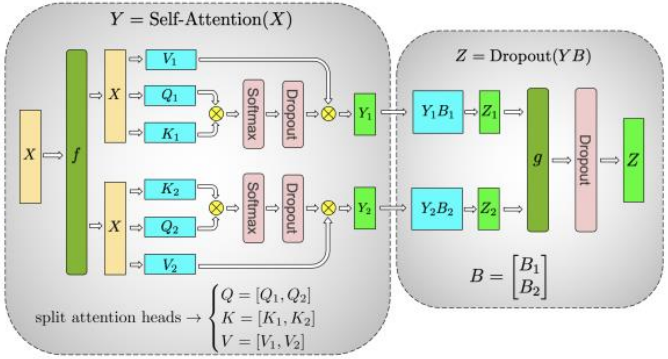




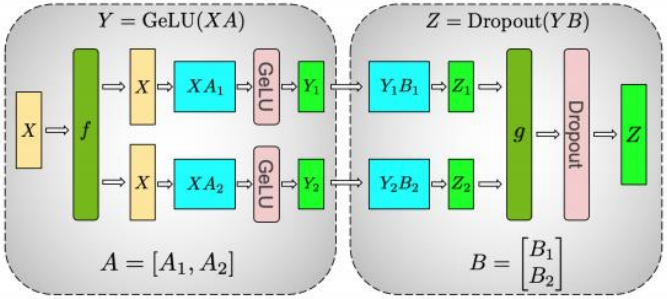
Tensor párhuzamosság

Megvalósítás:

- PyTorch: **torch.distributed.tensor.parallel** (experimental)
- **nanotron**, **megatron**



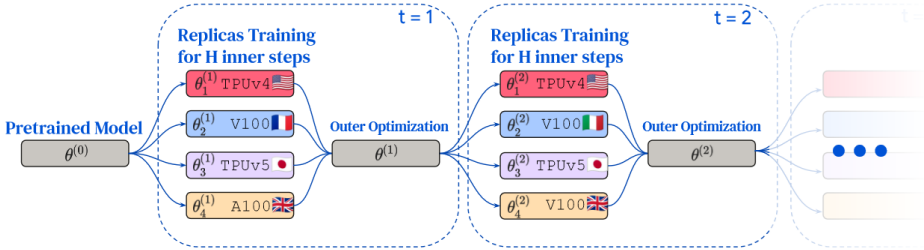
(b) Self-Attention



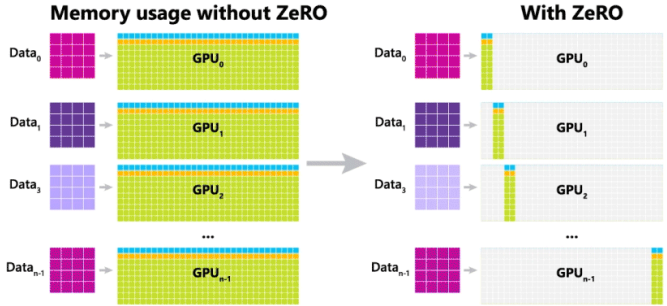
(a) MLP

Optimalizáló párhuzamosság

- DiLoCo (**opendiloco**)



- ZeRO stage-1 (**deepspeed**)



Milyen lehetőségeink vannak a megvalósításra?

```
pytorch
deepspeed - github.com/microsoft/DeepSpeed
megatron  - github.com/NVIDIA/Megatron-LM
nanotron  - github.com/huggingface/nanotron
pipegoose - github.com/xrsrke/pipegoose
```

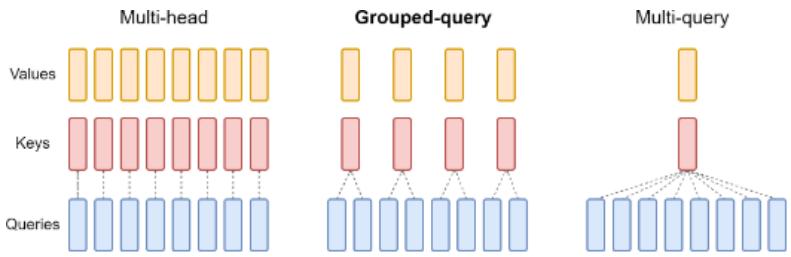
Egyéb Lehetőségek

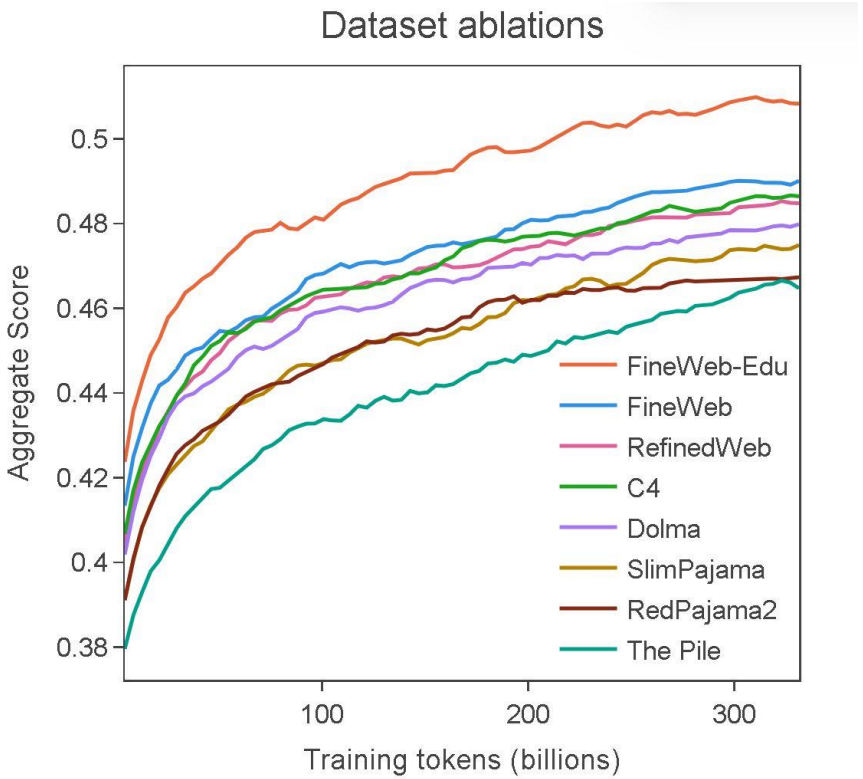
Egyéb Lehetőségek

- Optimizer Fusion
- Kvantálás tovább csökkentheti a model méretét és sebességét, a precizitásért cserébe, mind inferenci és tanítás során
  - **(Auto)GPTQ**, **AWQ**, **TensorRT**, **llama.cpp**…
- Ritkaság (!!! egyenlőre kerüljük)
  - A reprezentáció általában nem ritka, hanem sűrű és maszkolással van megoldva
  - Memória használat növekszik a maszkok beiktatása miatt
  - Továbbá a futás idő sem kedvező
  - Valós ritka reprezentáció esetén is a kernel-ek magas ritkaságot és struktúrát követelnek meg

Egyéb Lehetőségek

- Grouped Query Attention
- Multi Query Attention
- KV-sharing (across layers)
- vLLM: prefix-caching, chunked prefill
- …





Pytorch Conference 2024 - Making LLM training faster, Daniel Han

Köszönöm a figyelmet!