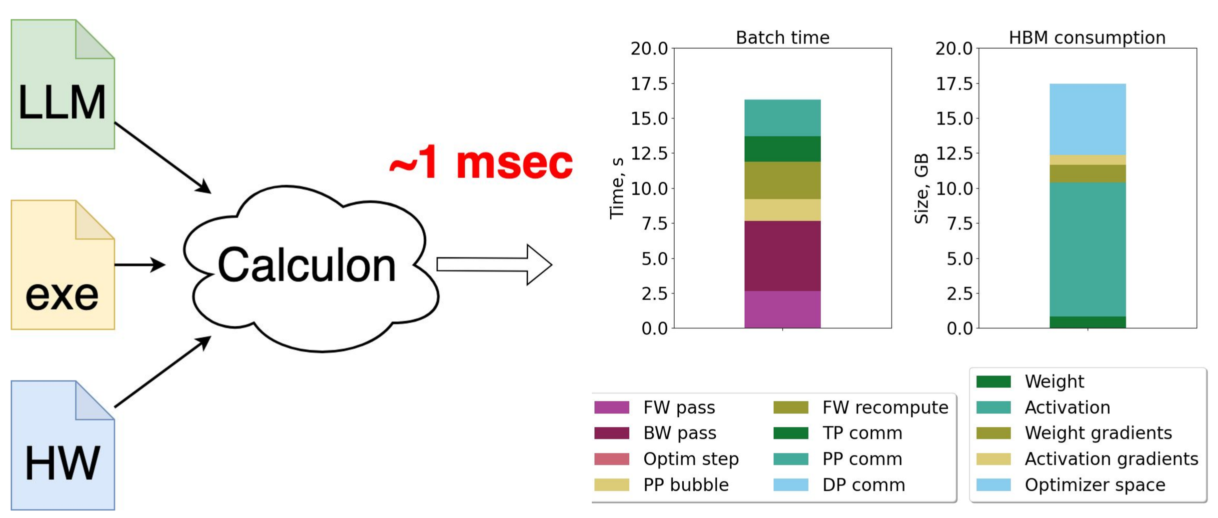
Scaling Infrastructure to Support Multi-Trillion Parameter LLM Training

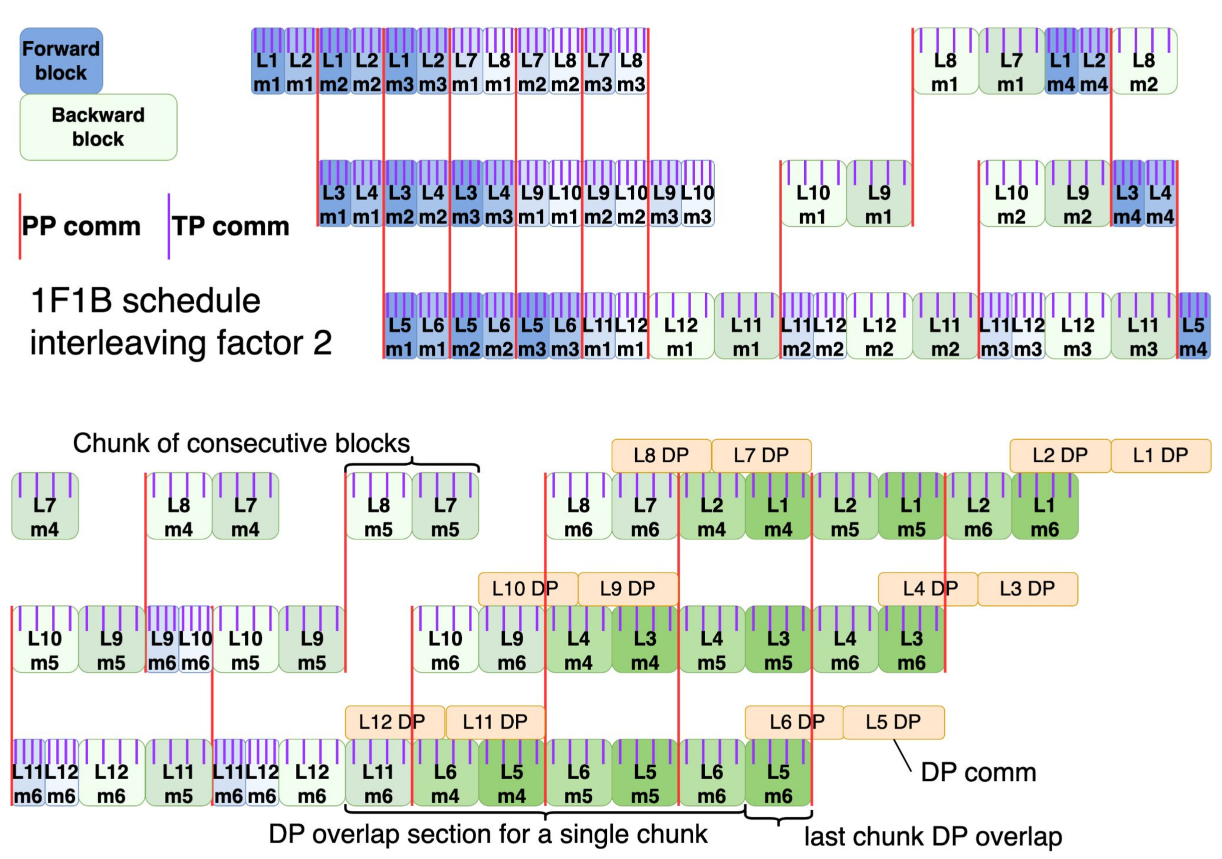
Mikhail Isaev[1], Nic McDonald[2], Rich Vuduc[1]
[1]Georgia Institute of Technology, [2]Nvidia

How we model LLMs

Calculon is a new **open source analytical model** for LLM co-design.
It takes three specifications: (a) **LLM architecture**; (b) LLM **execution strategy** defining parallelism and optimizations; and (c) **HW system** description.
It produces time and resource utilization **prediction**, in a format **similar to profiler** output



Calculon exploits **LLM's repetitive nature** – same computation repeats over again.
Just **six LLM parameters** define a single **Transformer block.**
LLM parameters and execution strategy builds an LLM execution DAG



How we scale LLMs

We can grow model width (hidden size), or model depth (number of layers).
Model ratio - a ratio between width and depth.
There is **no consensus** on how to scale LLMs.
We picked **linear ratio scaling** implied by the current LLM models.

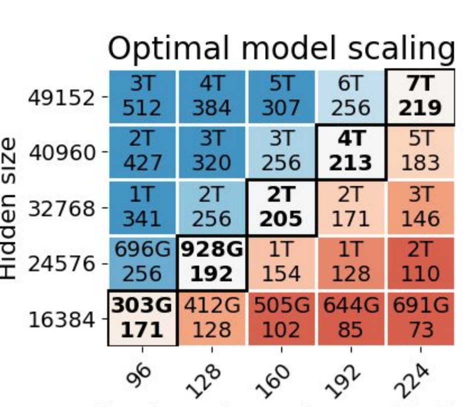


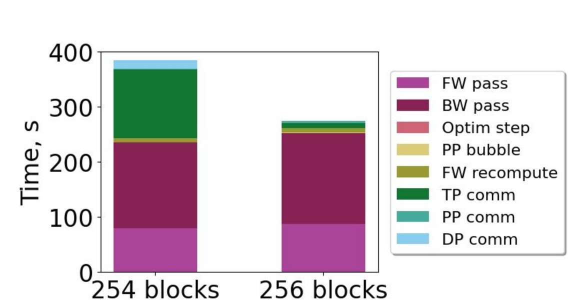
TABLE I
ASPECT RATIOS OF CURRENT LLMS.

Name	Hidden	# Blocks	Aspect Ratio
GPT2-1.5B [16]	1600	48	33.3
Jurassic-6.5B [10]	4096	32	128
PaLM-8B [3]	4096	32	128
GPT3-13B [2]	5140	40	128.5
Megatron-40B [11]	6144	40	153.6
PaLM-62B [3]	8192	64	128
Chinchilla-64B [4]	8192	80	102.4
GPT3-175B [2]	12288	96	128
Jurassic-175B [10]	13824	76	181.9
Megatron-309B [11]	16384	96	170.7
TuringNLG-530B [20]	20480	105	195
PaLM-540B [3]	18432	118	156
Megatron-1T [11]	25600	128	200

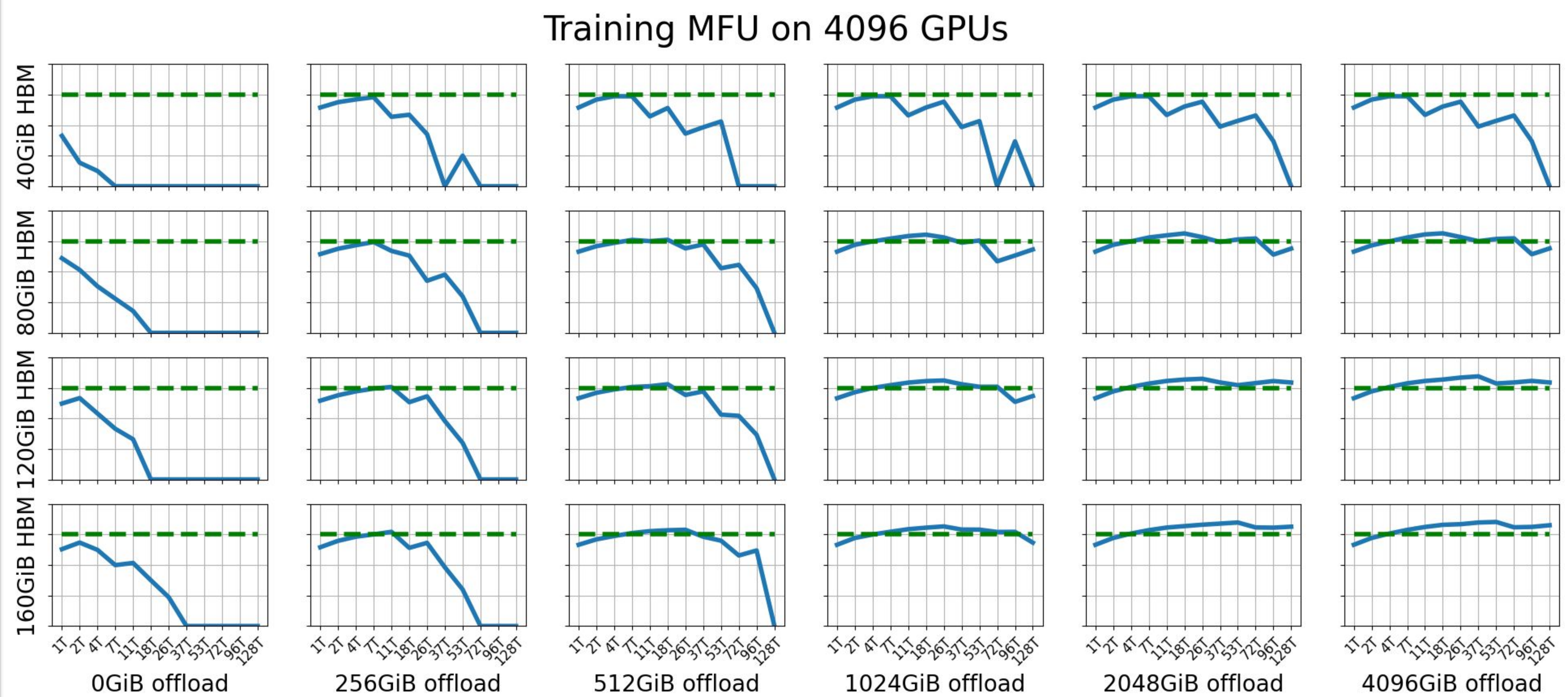
TABLE II
TWELVE MULTI-TRILLION PARAMETER LLMS, FROM 1T TO 128T.

Name	Hidden	# Blocks	Aspect Ratio
1T	24,576	128	192
2T	32,768	160	204.8
4T	40,960	192	213.3
7T	50,176	224	224
11T	60,416	256	236
18T	70,656	288	245
26T	81,920	320	256
37T	94,208	352	267.6
53T	106,496	384	277.3
72T	119,808	416	288
96T	134,144	448	299.4
128T	148,480	480	309.3

Model width and depth should change in the large power-of-two steps to provide best performance for tensor and pipeline parallelism

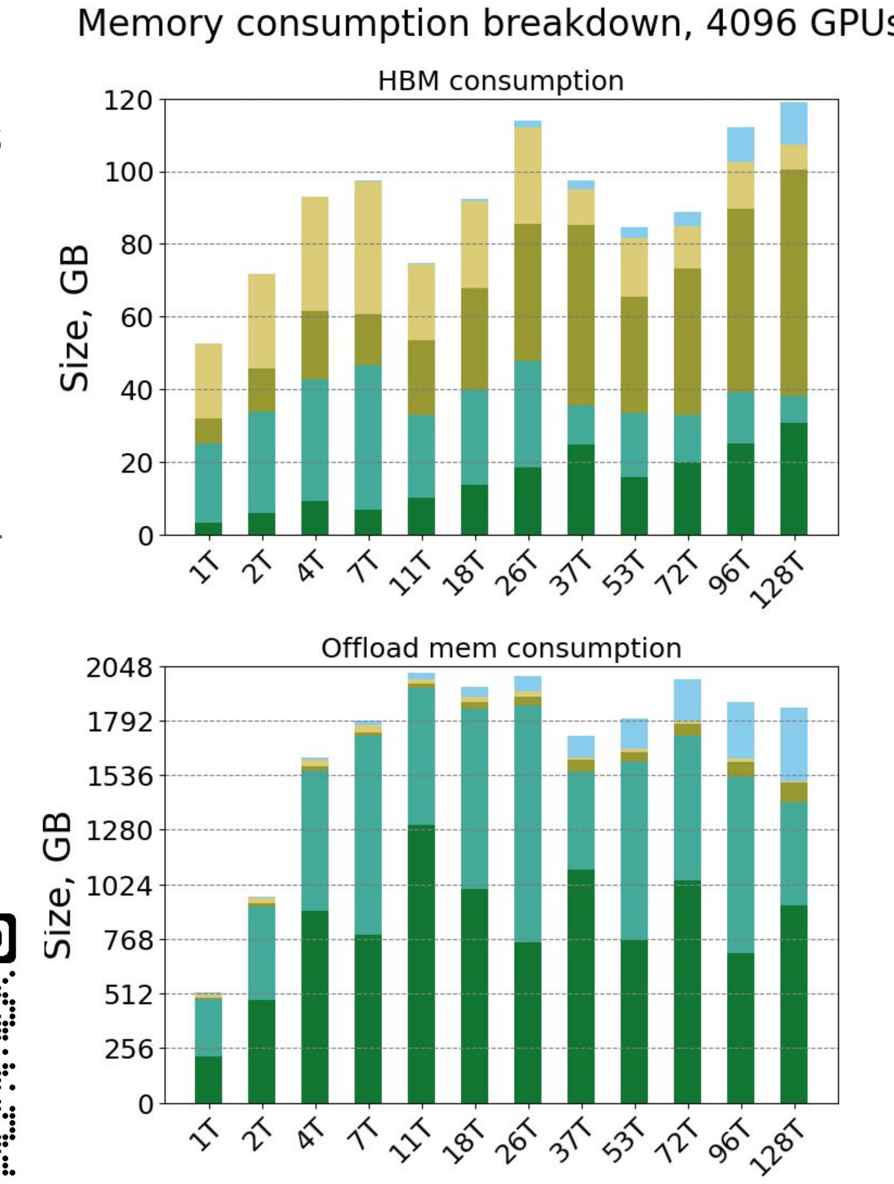


How can we train 100+T LLMs with 75% MFU?



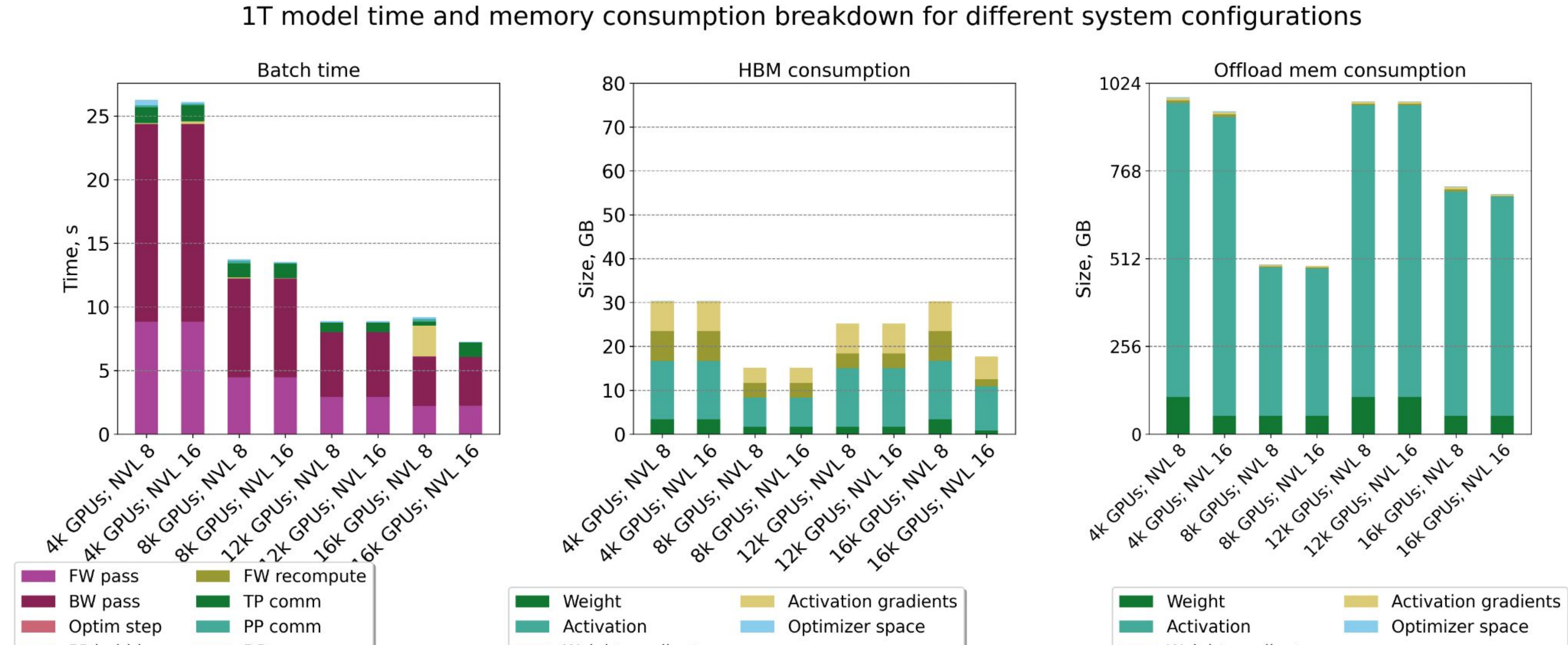
Key takeaways

1. Training a **100T**-parameters LLM takes **1 TiB** of offload memory per GPU with bidirectional bandwidth of **100 GB/s**
2. **Larger LLMs** need **larger HBM** even with offload memory
3. Small pipeline parallelism with **interleaved schedule helps** overlap **data parallel communication**
4. Smaller LLMs around **1T struggle to scale to 16k** GPUs due to various parallelism bottlenecks
5. **Smaller LLMs** utilize **larger data parallelism** due to fewer weights and gradients, smaller tensor parallel communication overlap; **batch size is the limiting** factor
6. Increasing **tensor parallelism** is **limited** by fast **network size, smaller** time for communication **overlap**, **lower efficiency** of matrix multiplication



This analysis is made possible with **Calculon**, an **open sourced LLM analytical model** for fast HW/SW co-design for LLM training
Find us at **https://github.com/calculon-ai**

1T model time and memory consumption breakdown for different system configurations



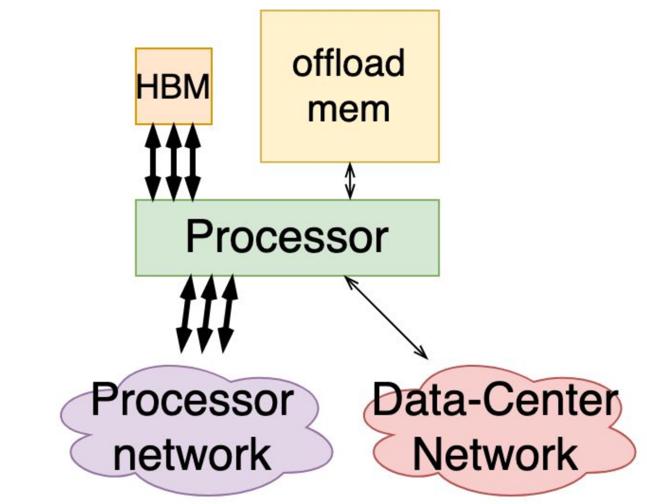
How tensor offloading works

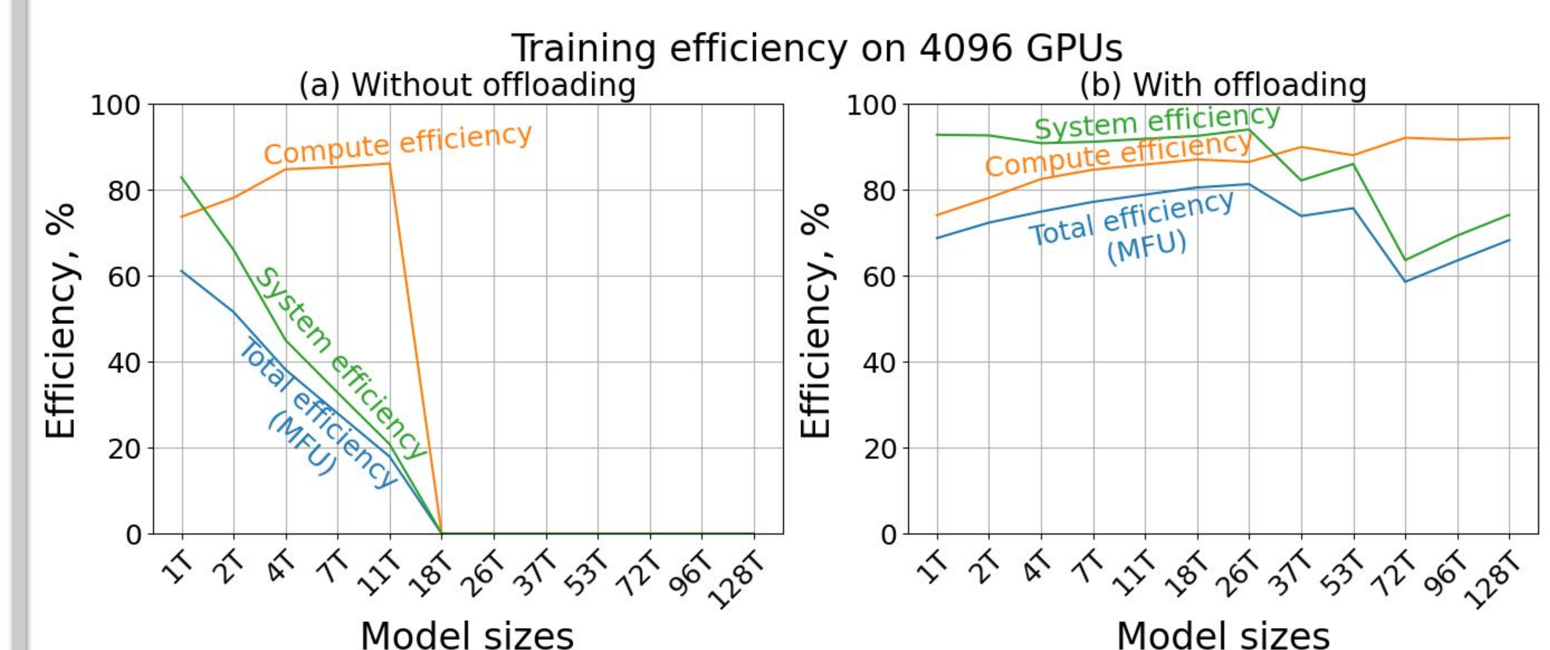
Fast memory with **lower capacity** only keeps a few **currently used** transformer **blocks**.

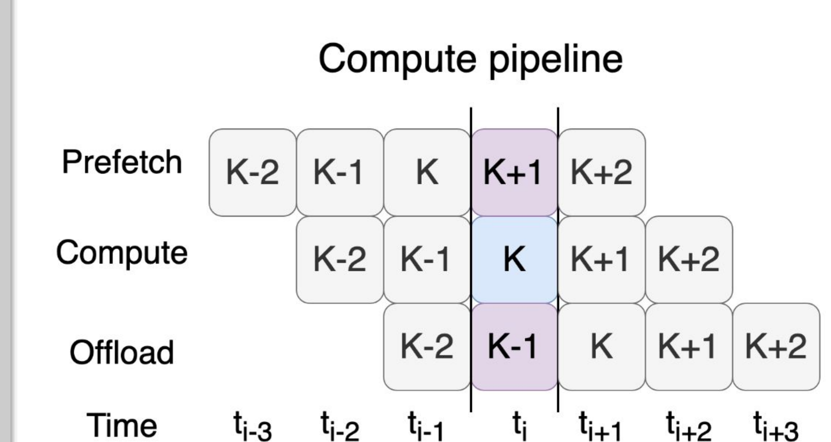
Rest is offloaded to slower memory.

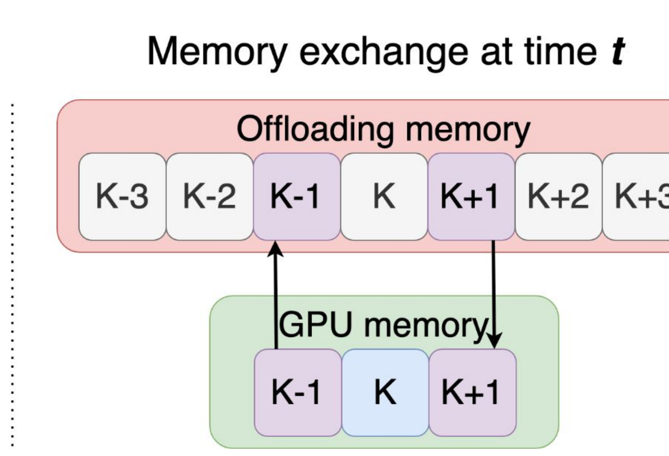
No overhead if enough time to **prefetch** tensors for the **next layer** and **offload** tensors for the **previous layer** during **current** layer's **compute**.

That depends on **memory bandwidth** and **amount of compute**.









How big and how fast should offload memory be?

100 GB/s is enough for offloading, allowing to use ***CXL*** or ***eth*** attached memory.
1-2 TiB is enough for most applications on different scale.
Smaller systems need **more memory**, **smaller models** need **more bandwidth**.