CS 598
AI Efficiency: Systems and Algorithms
Overview & Key Challenges in LLMs Training Systems

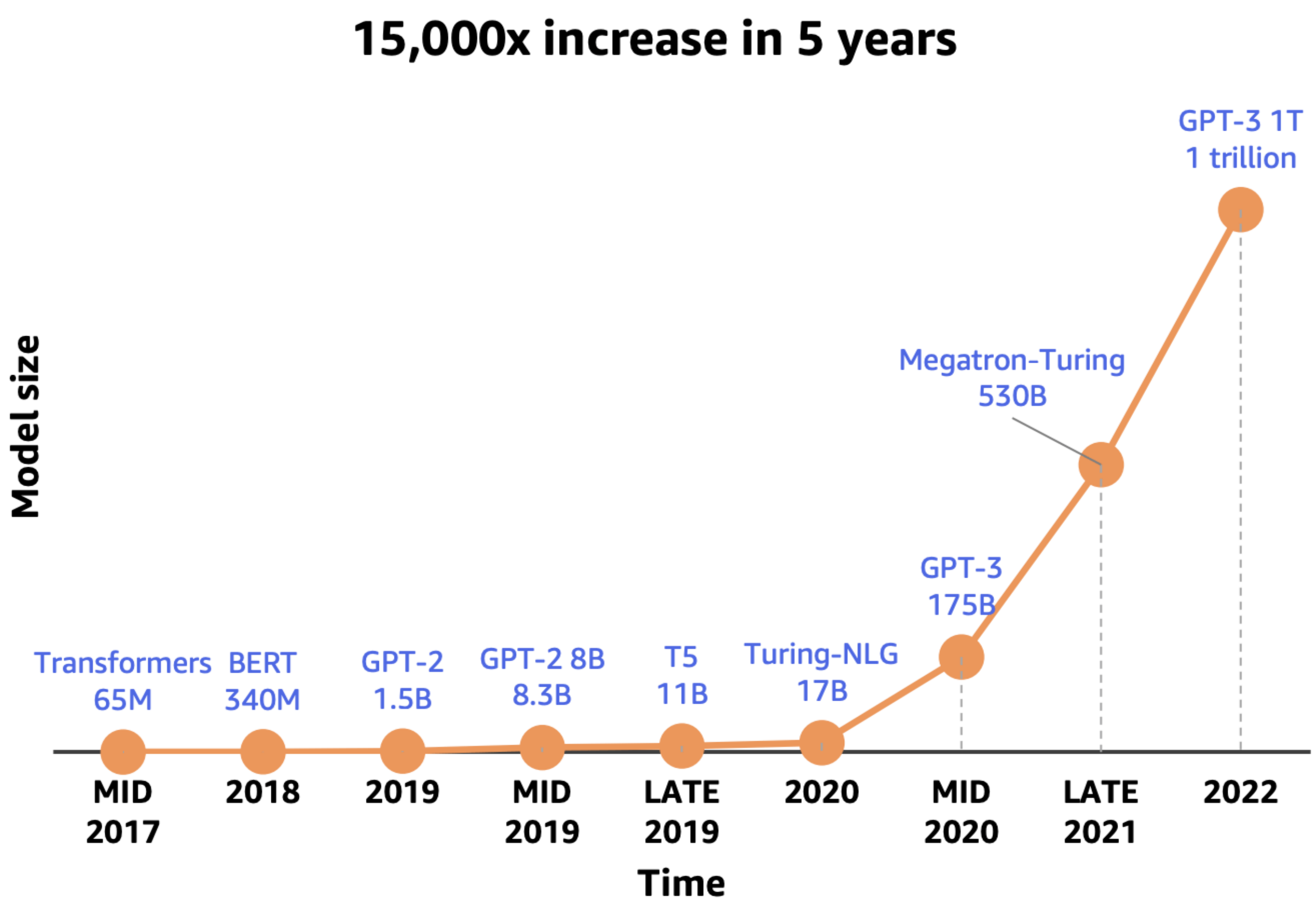
Minjia Zhang

Computer Science Department

The Large Language Model Revolution

What is Language Model?

Evolution of DNN Models



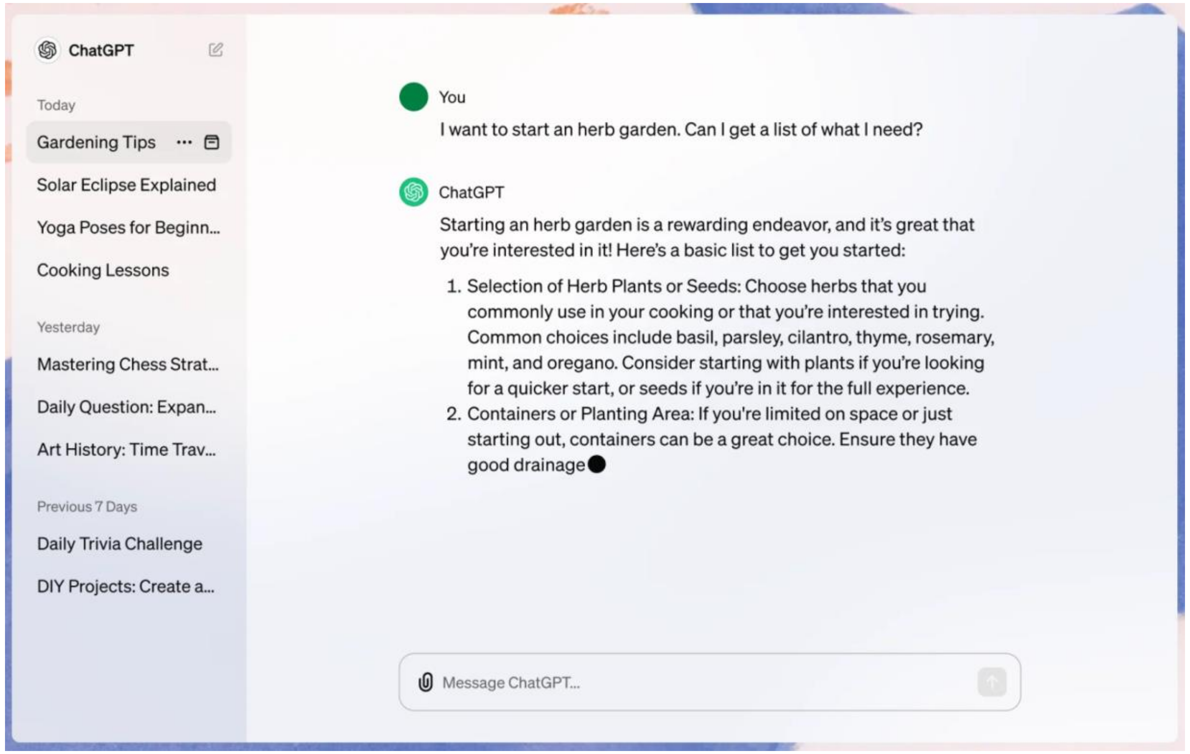
Larger models → better accuracy

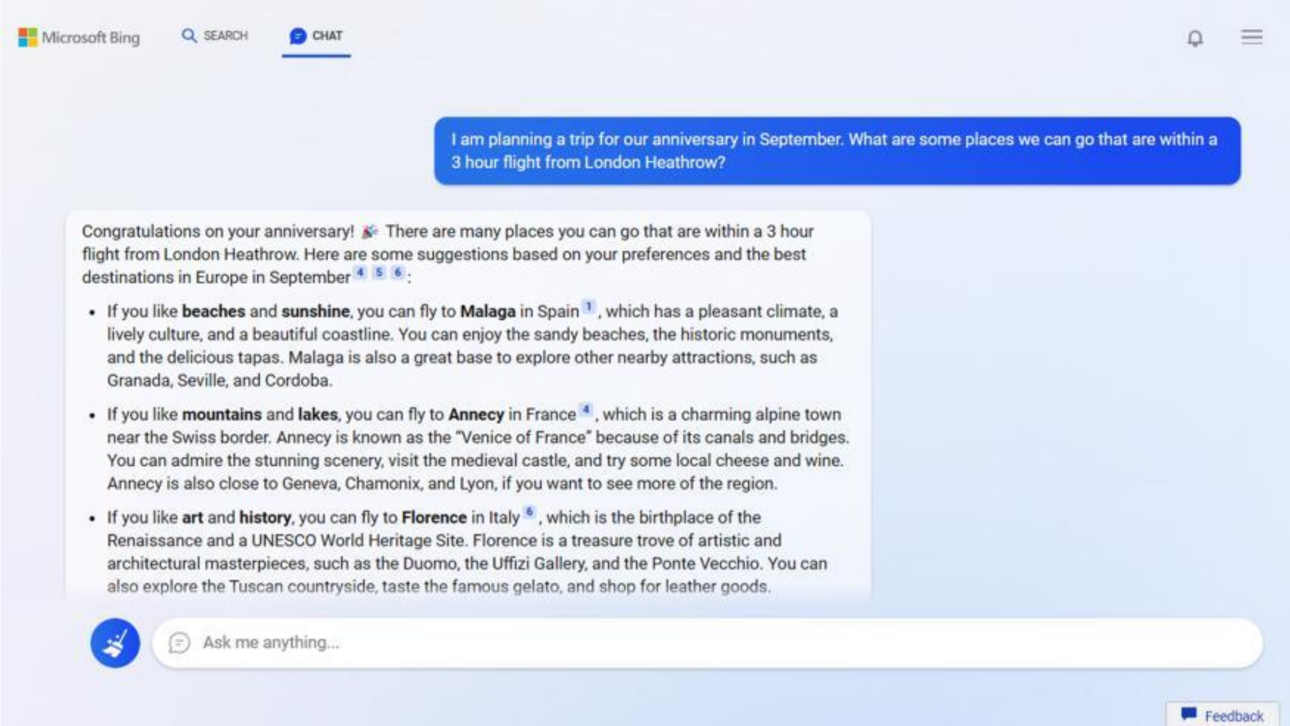
Model size is still growing

Not reached the accuracy limit yet

More compute-efficient to train larger models than smaller ones to same accuracy

Dialogue/New Search





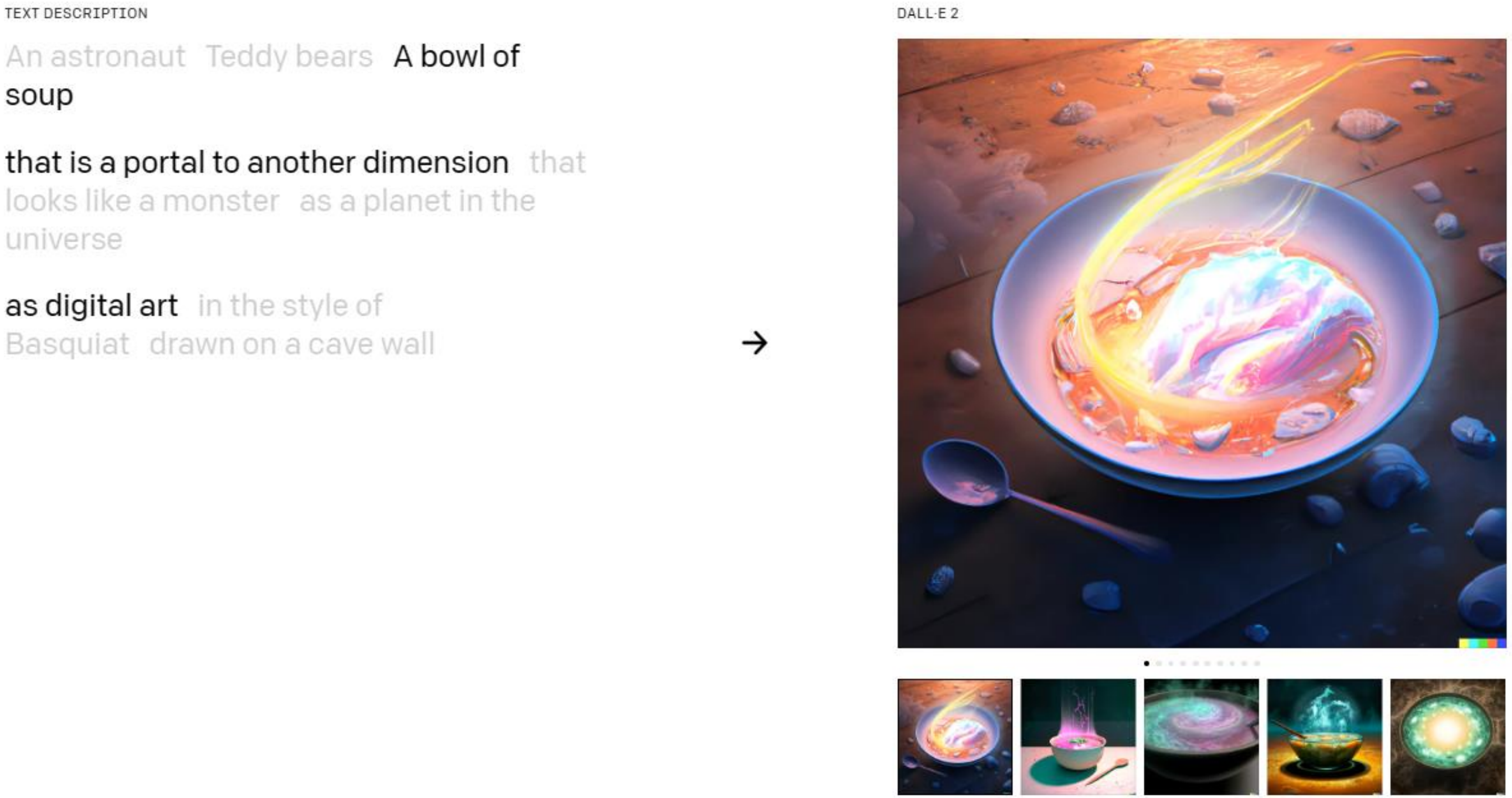
[ChatGPT: Optimizing Language Models for Dialogue]

Code Continuation and Generation

```python
"""
Python 3
Get the current value of a Bitcoin in US dollars using the bitcoincharts api
"""

import requests
import json

def get_bitcoin_price():
    url = 'http://api.bitcoincharts.com/v1/weighted_prices.json'
    response = requests.get(url)
    data = json.loads(response.text)
    return data['USD']['7d']

if __name__ == '__main__':
    print(get_bitcoin_price())
```

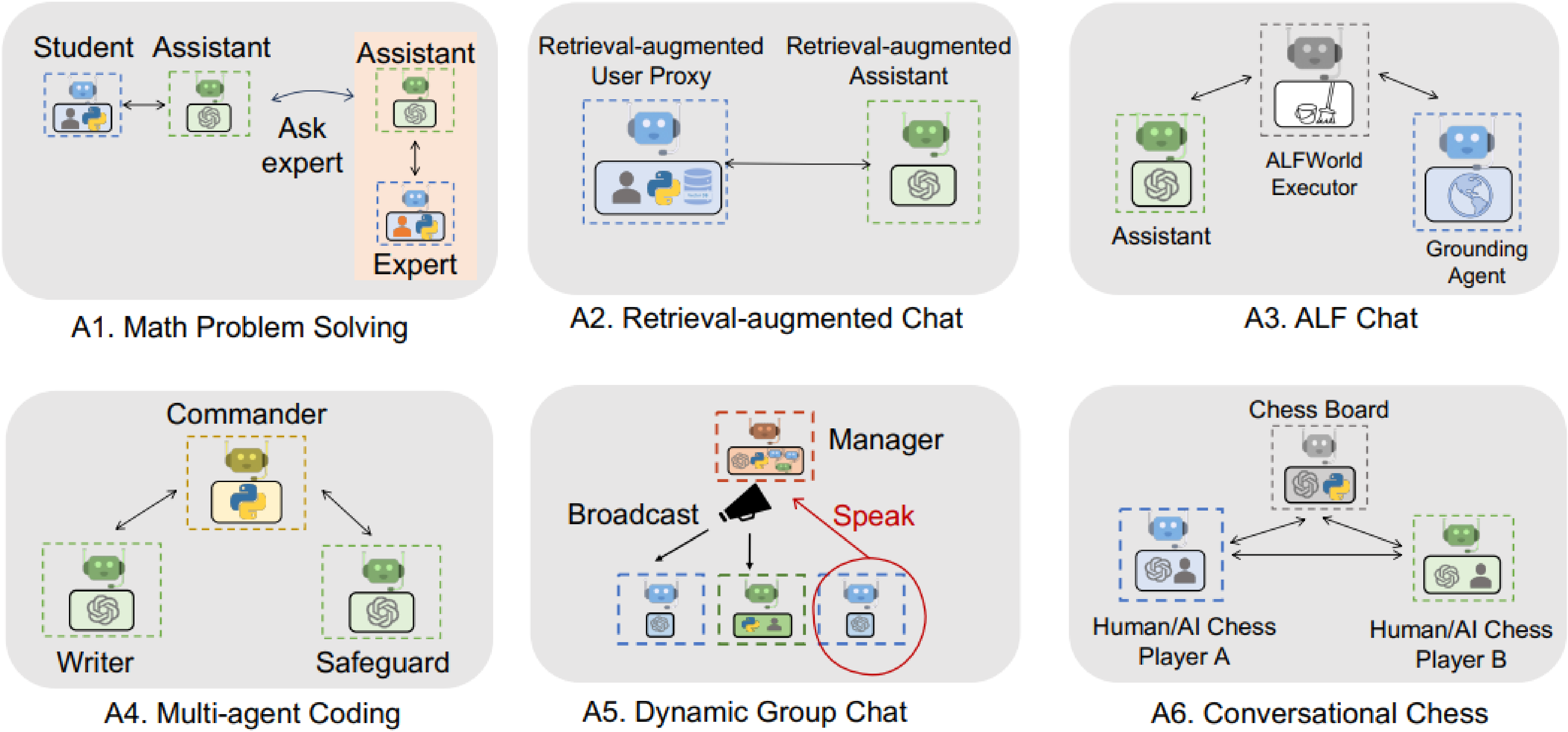
[Suggest code and entire function in your editor – Github/OpenAI Codex]()

Image Generation from Text



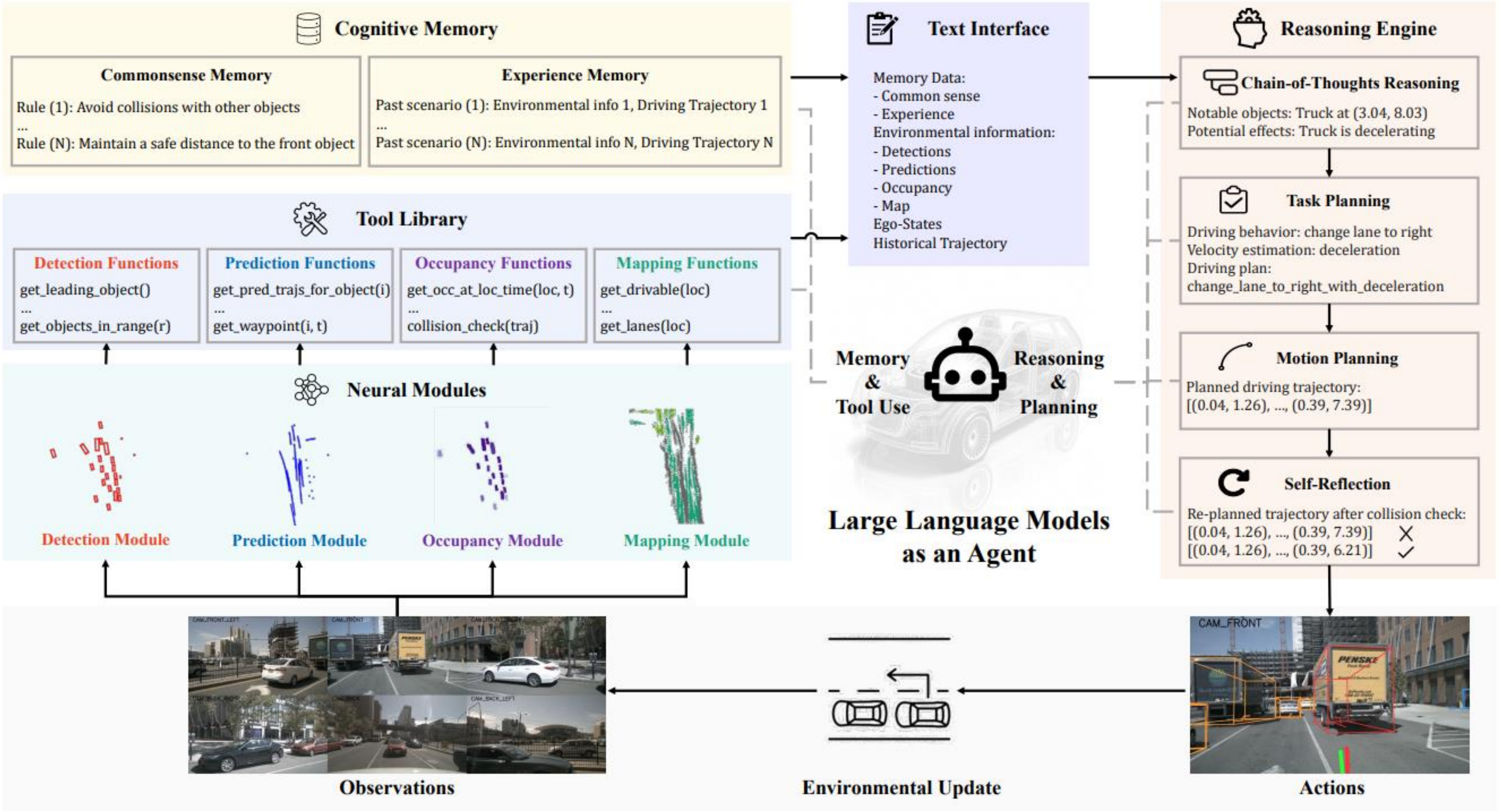
[DALL·E: Creating Images from Text - OpenAI]

Multi-Agent LLM Applications



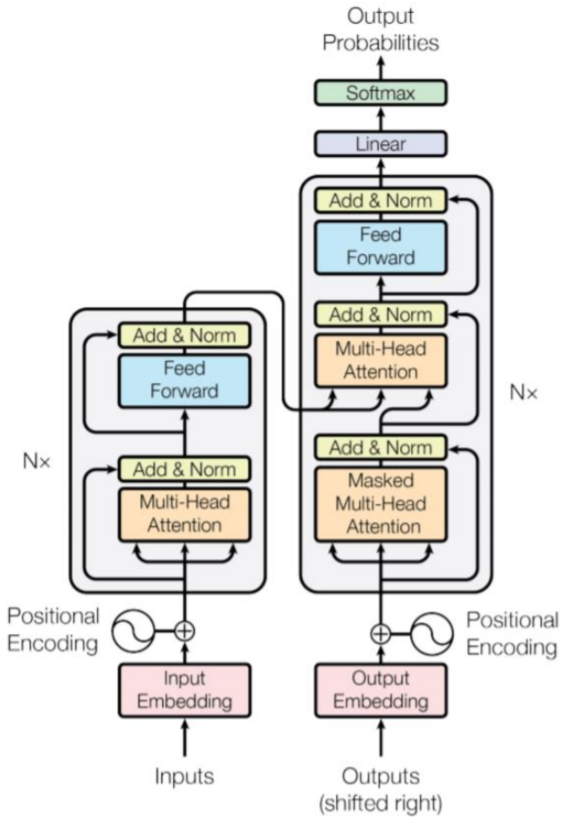
[AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation]

Autonomous Driving

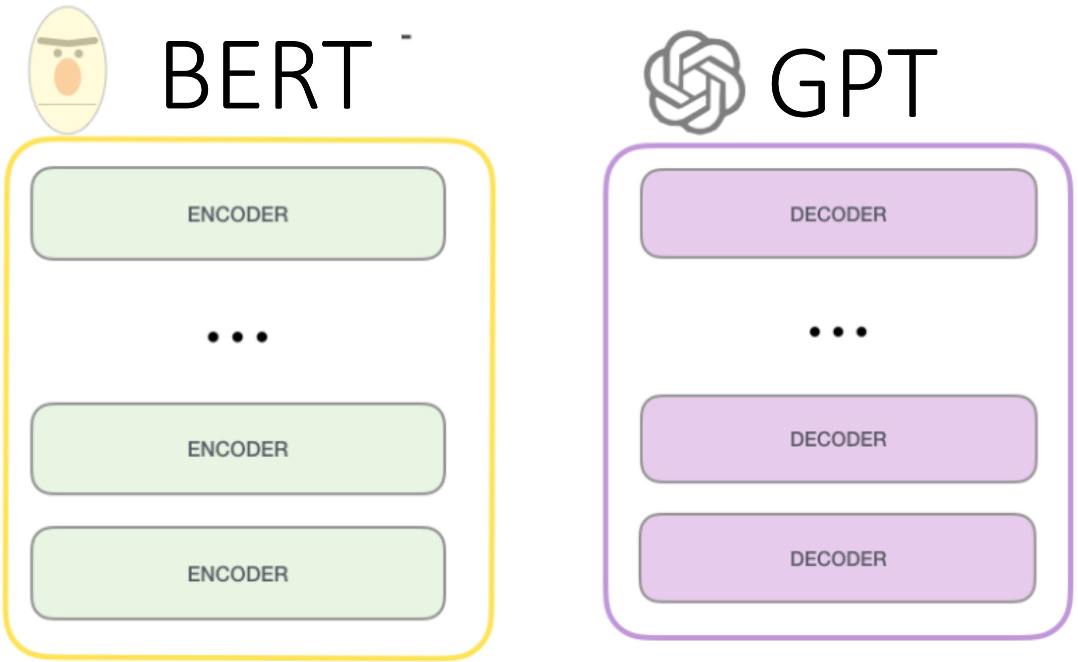


[A Language Agent for Autonomous Driving]()

Transformers for Language Modeling



Attention Is All You Need, NeurIPS 2017



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, ACL 2019

Language Models are Few-Shot Learners, NeurIPS 2020

LLMs are Impressively Scaling!

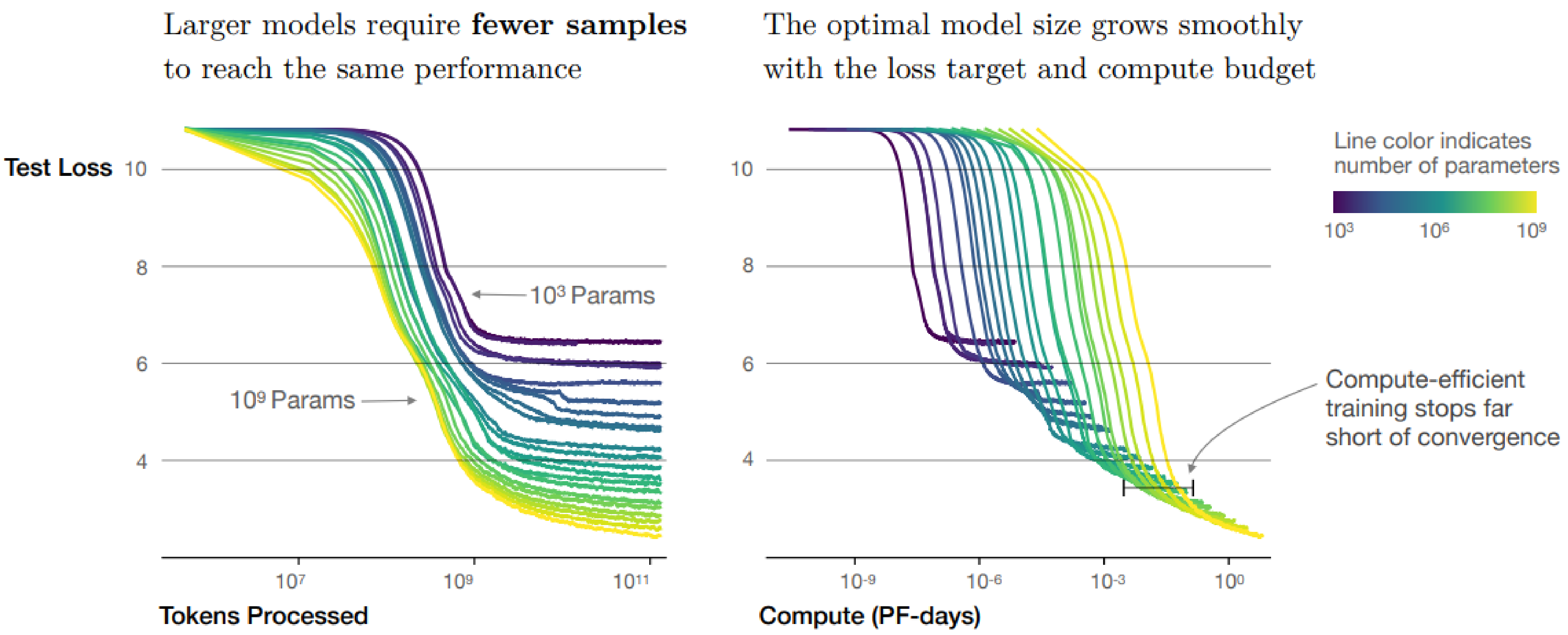


Figure 2 We show a series of language model training runs, with models ranging in size from 10^3 to 10^9 parameters (excluding embeddings).

Scaling Laws for Neural Language Models, OpenAI, 2020

AI Efficiency Challenges

- Too slow to train high-quality models on massive data
  - More hardware ≠ higher throughput, bigger model
  - Higher throughput ≠ better accuracy, faster convergence, lower cost
  - Better techniques ≠ handy to use

- Slow and expensive to deploy the models

DL System Desired Capabilities (3E)

Efficiency: Efficient use of hardware for high scalability and throughput

Effectiveness: High accuracy and fast convergence, lowering cost

Easy to use: Improve development productivity of model scientists

ML/DL Training Problem Definition Recap

- Given model f, data set $\{xi, y_i\}_{i=1}^{N}$
- Minimize the loss between predicted labels and true labels:

  Min $\frac{1}{N}\sum_{i=1}^{N} loss(f(x_i, y_i))$

- Common loss function
  - Cross-entropy, MSE (mean squared error)

- Common way to solve the minimization problem
  - Stochastic gradient descent (SGD)
  - Adaptive learning rates optimizers (e.g., Adam)

Gradient Descent

- Model f_w is parameterized by weight w
- η > 0 is the learning rate

Backward pass

Forward pass

For t = 1 to T

$\Delta w = \eta \times \frac{1}{N} \sum_{i=1}^{N} \boxed{\nabla} \left(\boxed{loss(f_w(x_i, y_i))} \right)$ // compute derivative and update

w -= Δw // apply update

End

Adaptive Learning Rates (Adam)

- Model f_w is parameterized by weight w
- $\eta > 0$ is the learning rate

For t = 1 to T

$$\Delta w = \eta \times \frac{1}{N} \sum_{i=1}^{N} \nabla \left(loss\left(f_w(x_i, y_i)\right)\right)$$

$w \mathrel{-}= \boxed{\Delta w}$ // apply update

End

$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$
$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$
$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$

g_t : *Gradient at time t along* ω^j

ν_t : *Exponential Average of gradients along* ω_j

s_t : *Exponential Average of squares of gradients along* ω_j

β_1, β_2 : *Hyperparameters*

Adam: A Method for Stochastic Optimization, 2014

Accelerating Gradient Descent

- Model f_w is parameterized by weight w
- $\eta > 0$ is the learning rate

Can we accelerate it?

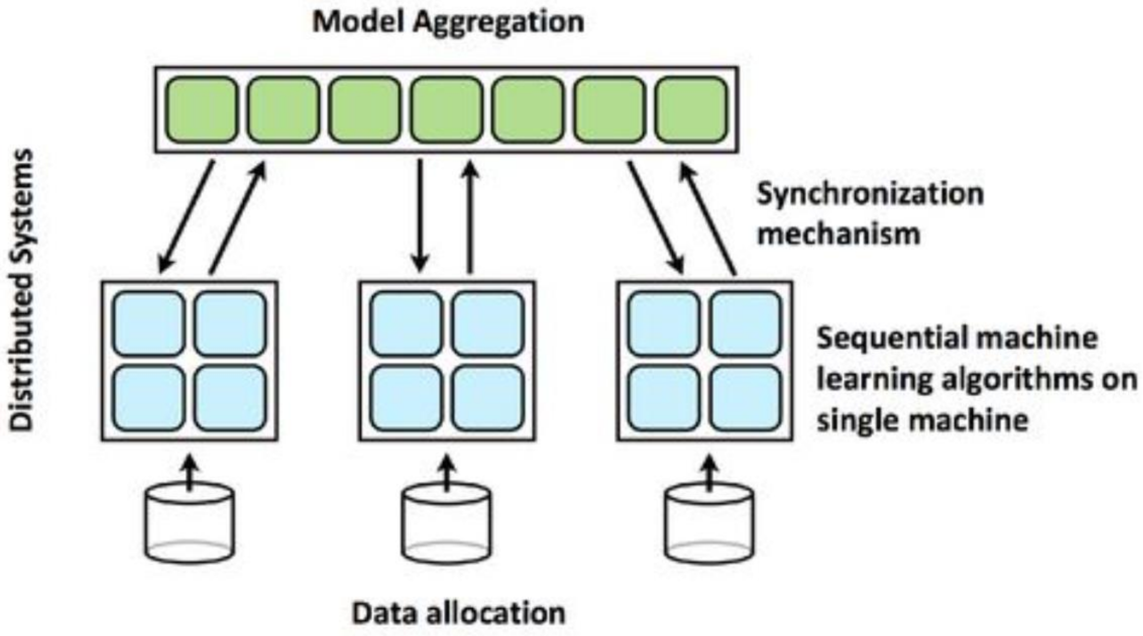
For t = 1 to T

$$\boxed{\Delta w = \eta \times \frac{1}{N} \sum_{i=1}^{N} \nabla \left(loss(f_w(x_i, y_i)) \right)}$$ // compute derivative and update

w -= Δw // apply update

End

Data Parallelism (DP)

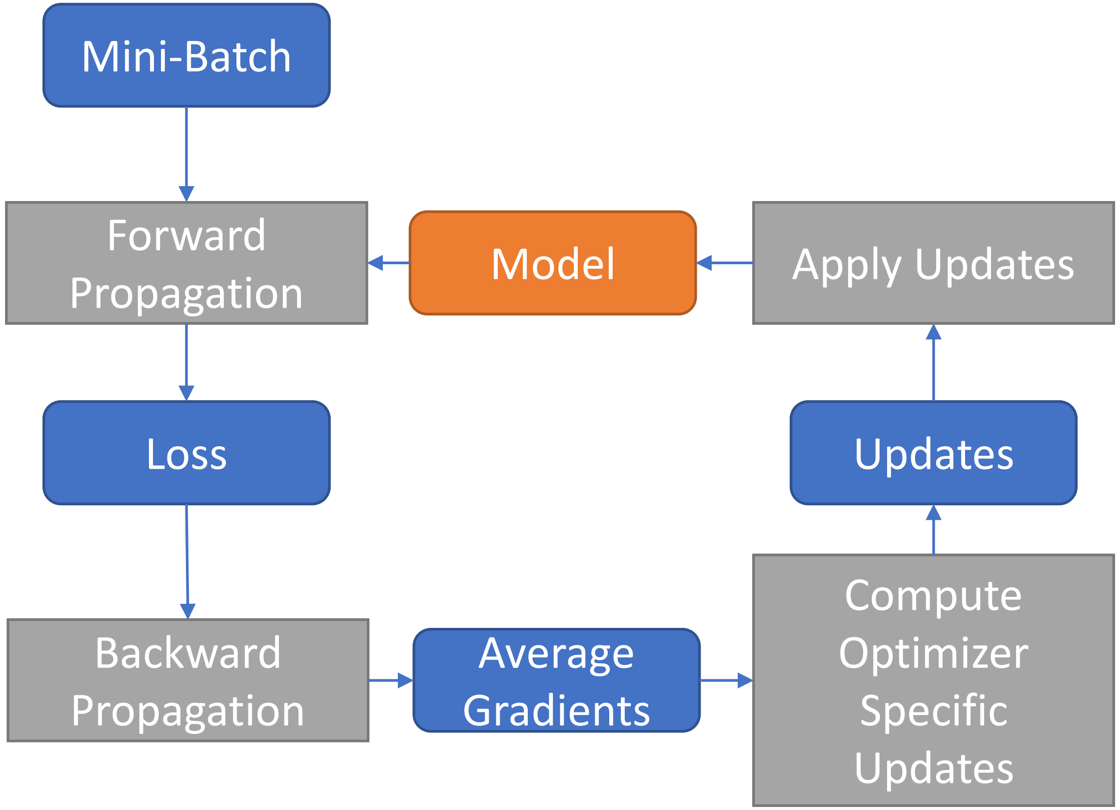


Implemented as standard component in DL training frameworks, such as PyTorch DDP

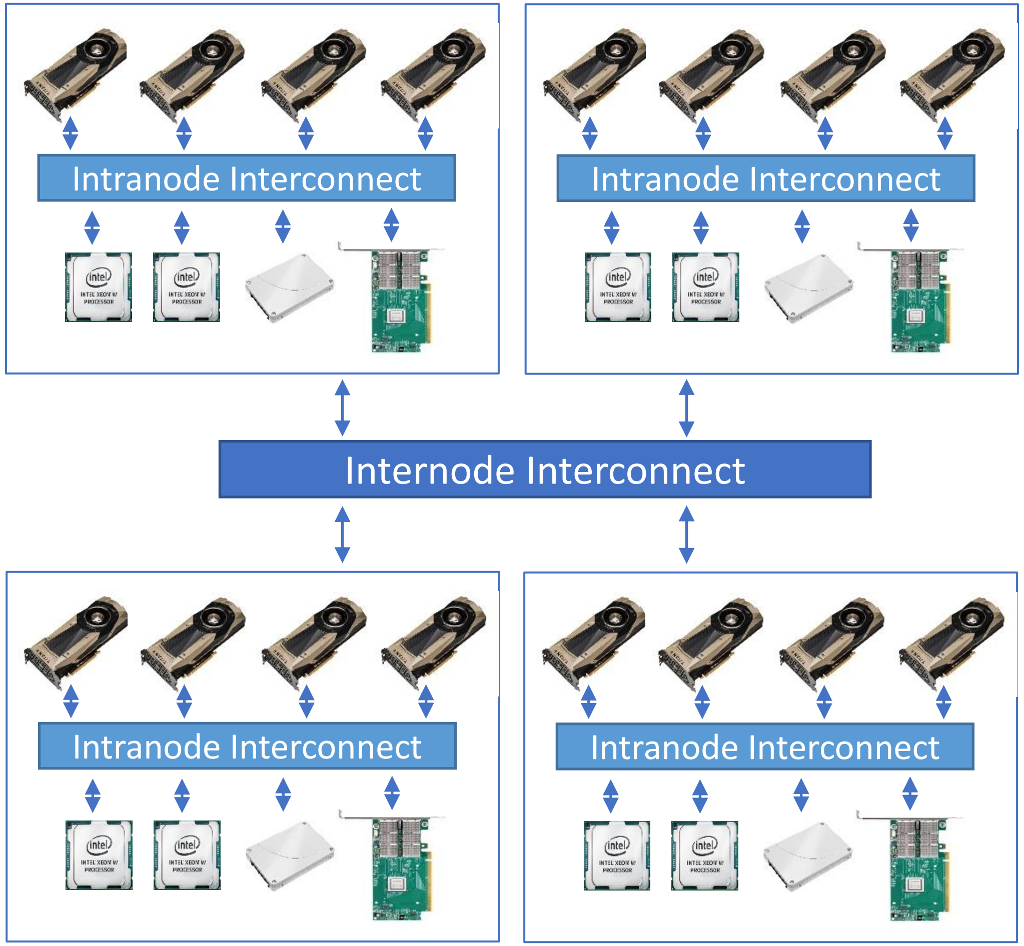
1. Partition the training data
2. Parallel training on different machines
3. Synchronize the local updates
4. Refresh local model with new parameters, then go to 2

Scaling Distributed Machine Learning with the Parameter Server, 2014

Distributed Data Parallel Training in GPU Clusters



Data Parallel Training Loop

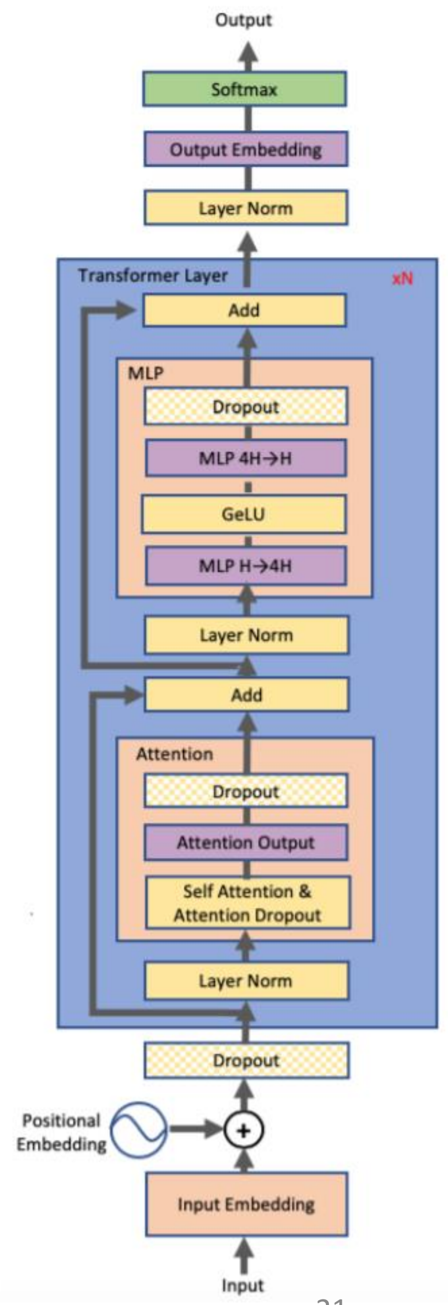


Distributed GPU Cluster

Training Efficiency: Breaking the Memory Wall

Large Model Training Challenges

	Bert-Large	GPT-2	Turing 17.2 NLG	GPT-3
Parameters	0.32B	1.5B	17.2B	175B
Layers	24	48	78	96
Hidden Dimension	1024	1600	4256	12288
Relative Computation	1x	4.7x	**54x**	**547x**
Memory Footprint	5.12GB	24GB	**275GB**	**2800GB**

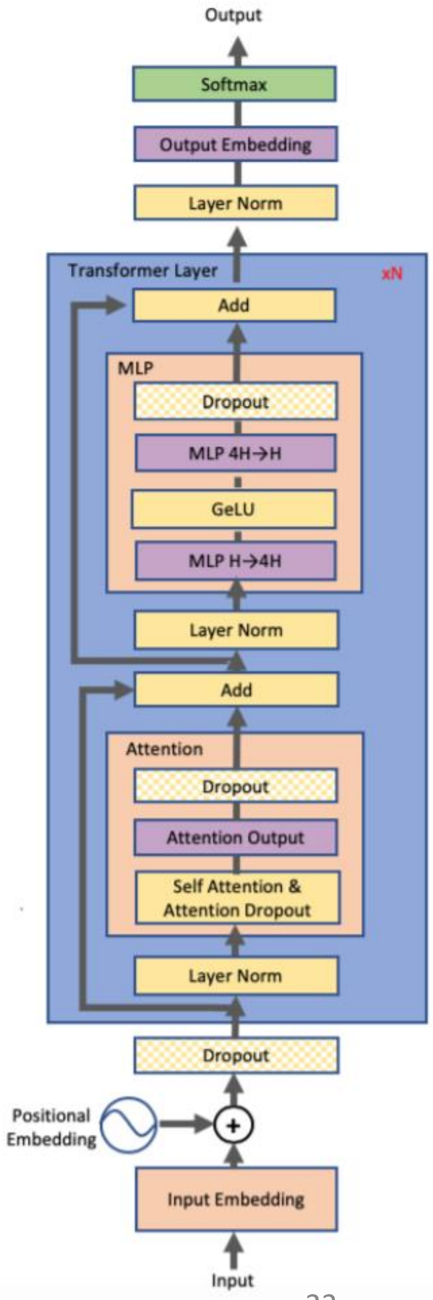


Large Model Training Challenges

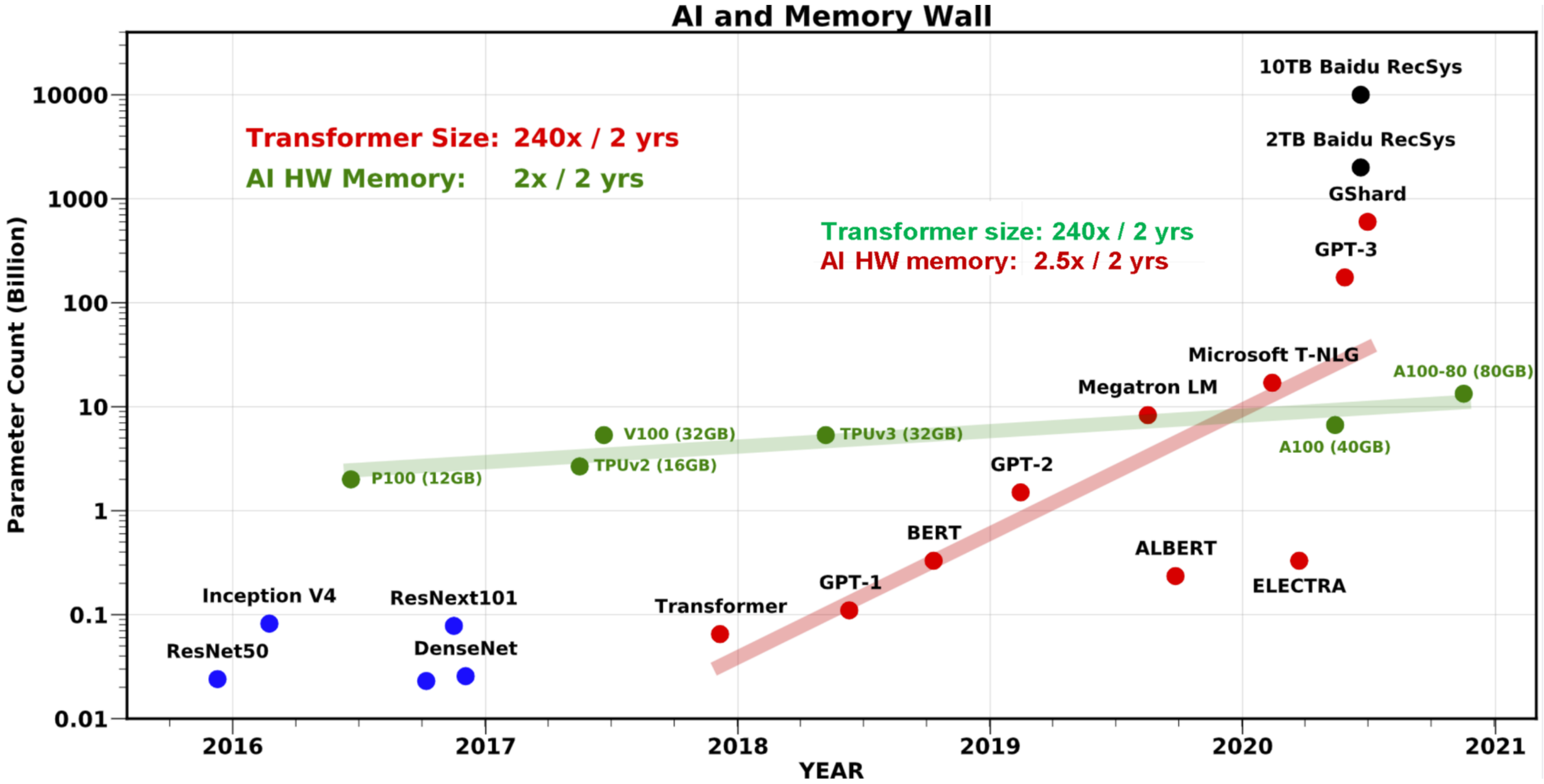
	Bert-Large	GPT-2	Turing 17.2 NLG	GPT-3
Parameters	0.32B	1.5B	17.2B	175B
Layers	24	48	78	96
Hidden Dimension	1024	1600	4256	12288
Relative Computation	1x	4.7x	**54x**	**547x**
Memory Footprint	5.12GB	24GB	**275GB**	**2800GB**

Out of Memory

NVIDIA V100 GPU memory capacity: 16G/32G
NVIDIA A100 GPU memory capacity: 40G/80G



DNN Training Hits the Memory Wall



*AI and Memory Wall. (This blogpost has been written in… | by Amir Gholami | riselab | Medium

Distributed Training Techniques

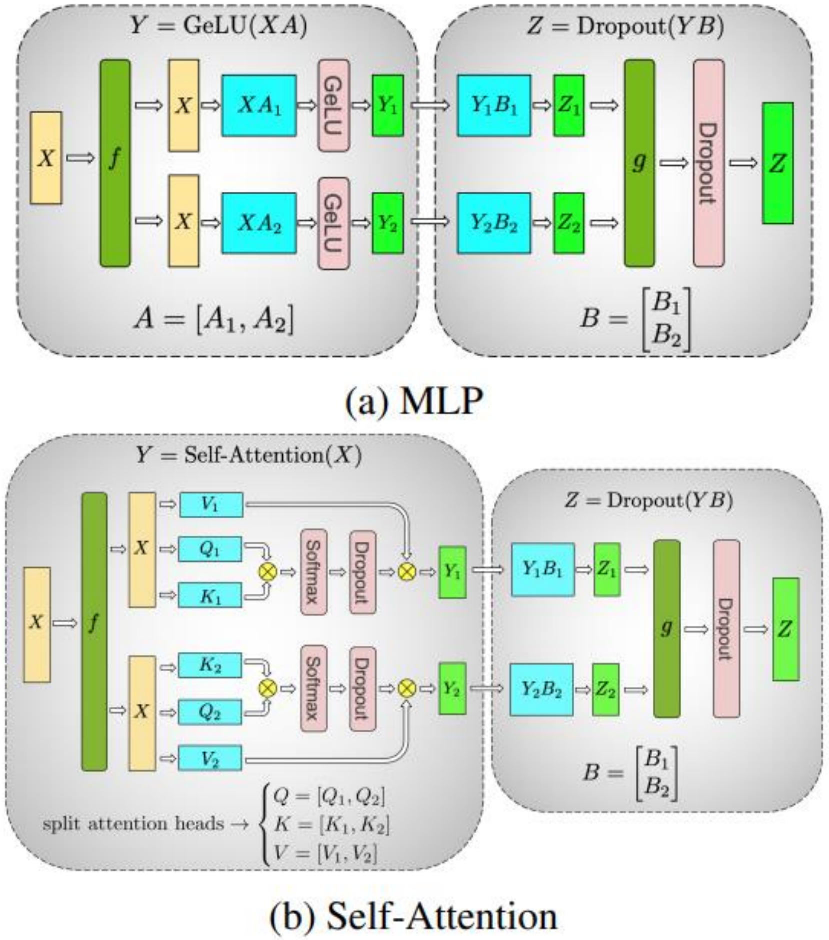
- Tensor Parallelism
- Pipeline Parallelism
- 3D Parallelism
- ZeRO-Style Data Parallelism
- …

Tensor Parallelism

Splice tensors across GPUs

+

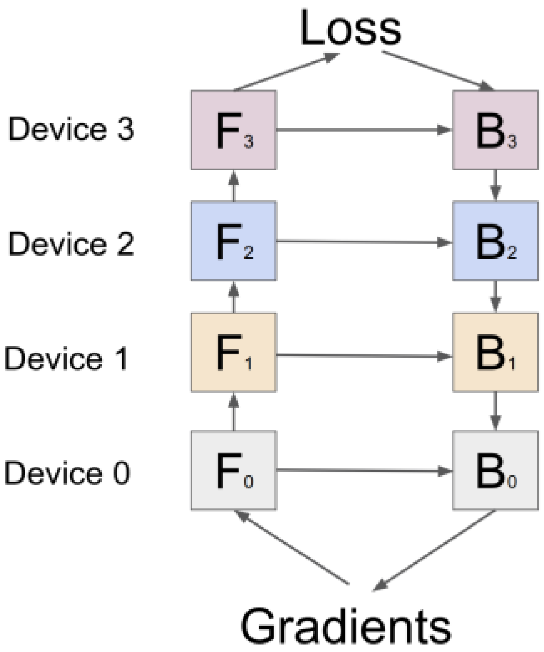
synchronization primitives (e.g., all-reduce)

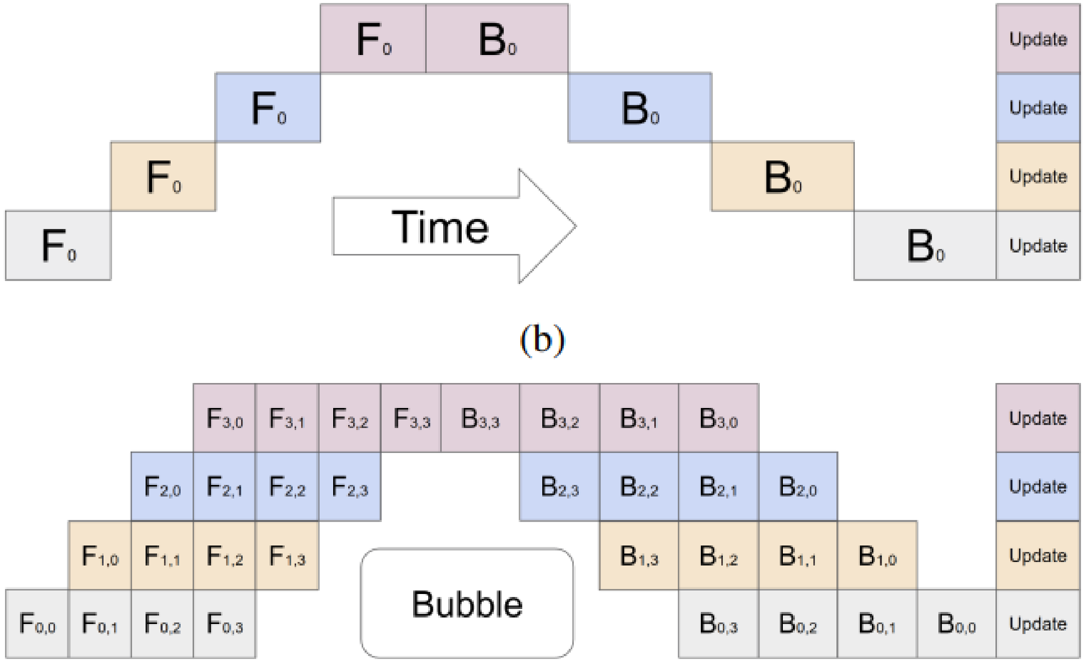


Supported in:
- [DeepSpeed]
- [Megatron-LM]

Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, 2019

Pipeline Parallelism





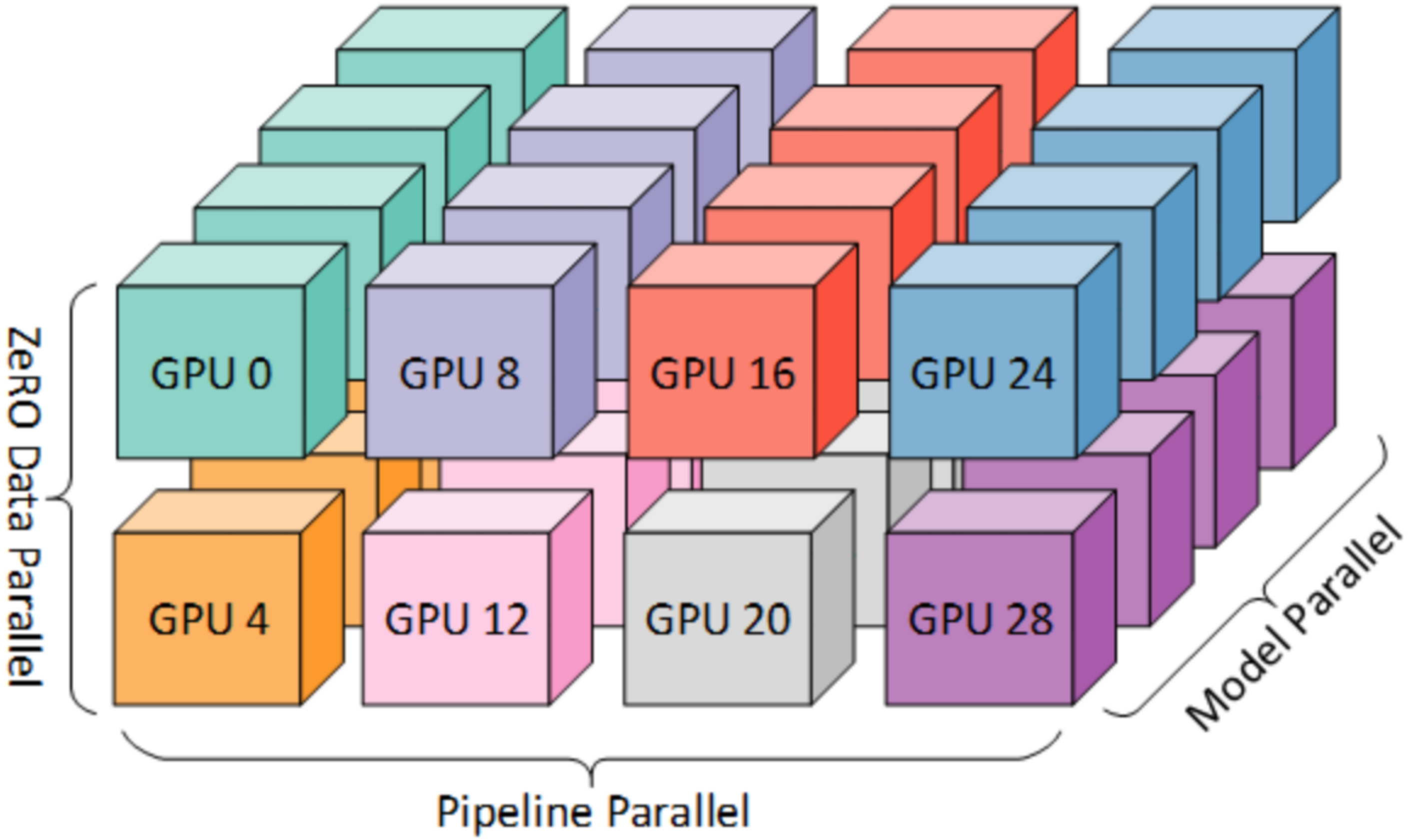
Supported in:

- PyTorch
- DeepSpeed
- Megatron-LM

- Naïve model parallelism leads to severe underutilization
- Gpipe divides batch into micro-batches, enabling different device to work on different micro-batches, reducing pipeline bubbles and improving utilization

GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism, 2019

3D Parallelism

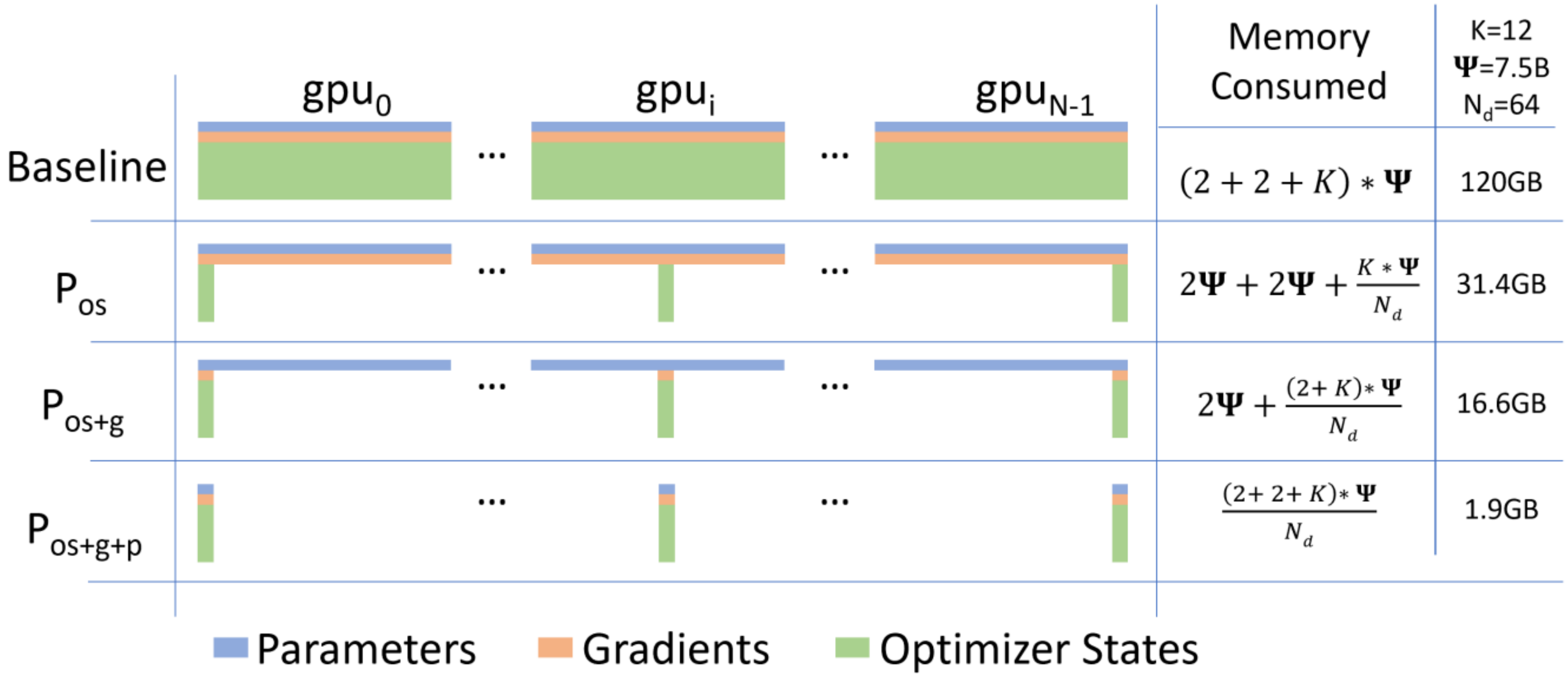


Supported in:
- [DeepSpeed]
- [Megatron-LM]

DeepSpeed Extreme Scale Model Training For Everyone, 2020

ZeRO-Style Data Parallelism

- ZeRO removes the redundancy across data parallel process
- Partitioning optimizer states, gradients and parameters (3 stages)



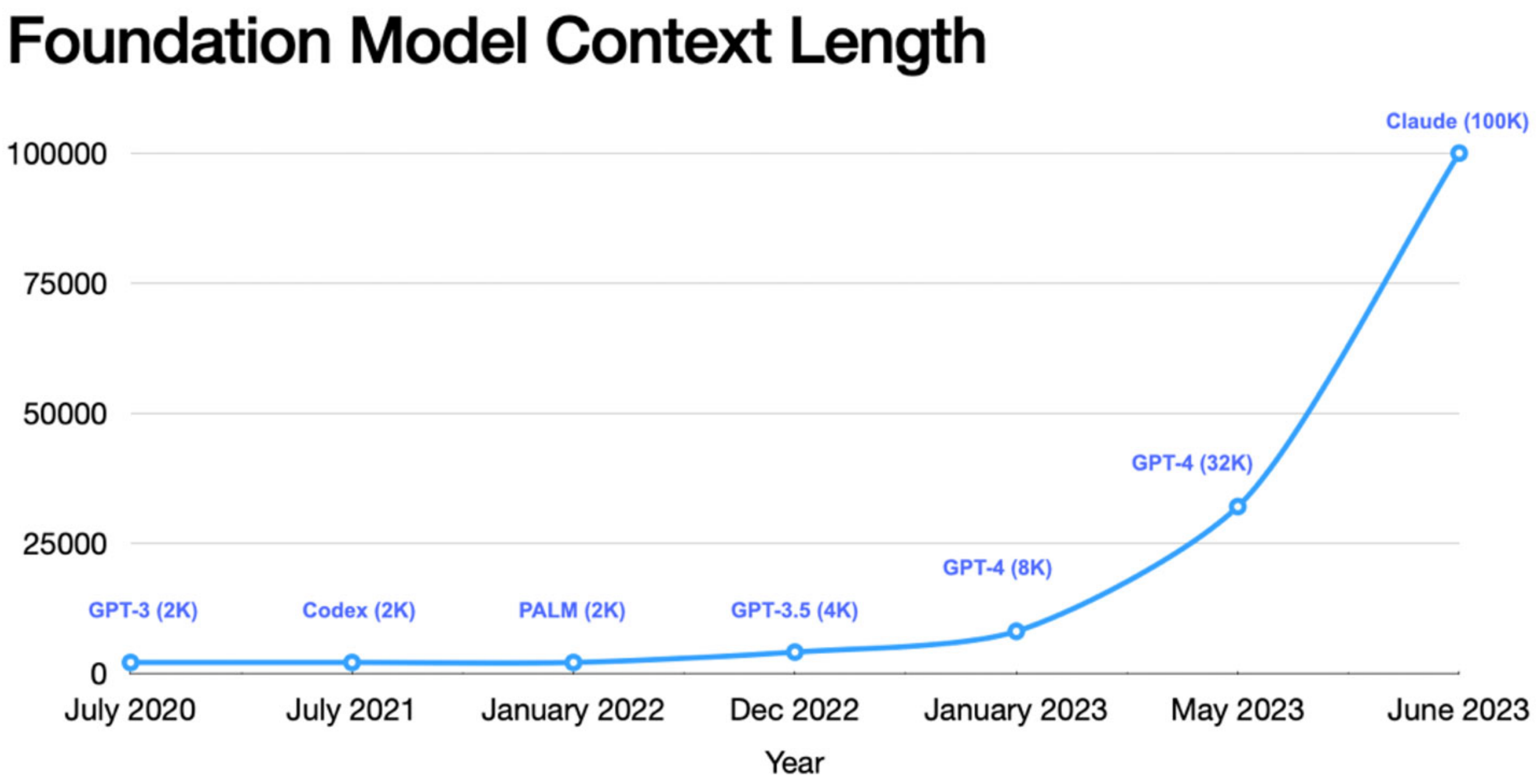
Supported in:
- DeepSpeed
- PyTorch

`ZeRO: Memory Optimizations Toward Training Trillion Parameter Models, 2019`

Large Models Need Parallelism

	Max Parameter (in billions)	Max Parallelism	Compute Efficiency	Usability (Model Rewrite)
Data Parallel (DP)	Approx. 1.2	>1000	Very Good	Great
Tensor Parallel (TP)	Approx. 20	Approx. 16	Good	Needs Model Rewrite
TP + DP	Approx. 20	> 1000	Good	Needs Model Rewrite
Pipeline Parallel (PP)	Approx. 100	Approx. 128	Very Good	Needs Model Rewrite
PP + DP	Approx. 100	> 1000	Very Good	Needs Model Rewrite
TP + PP + DP	> 1000	> 1000	Very Good	Needs Significant Model Rewrite
ZeRO	*> 1000*	*> 1000*	Very Good	Great

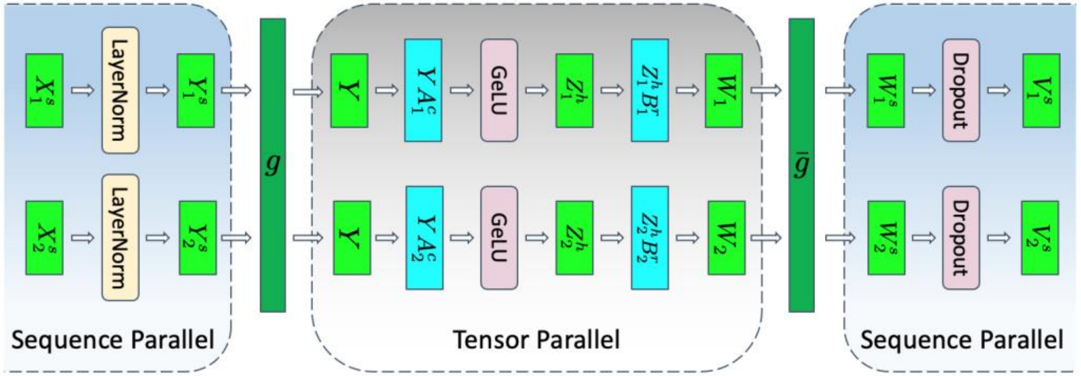
Long Context LLMs

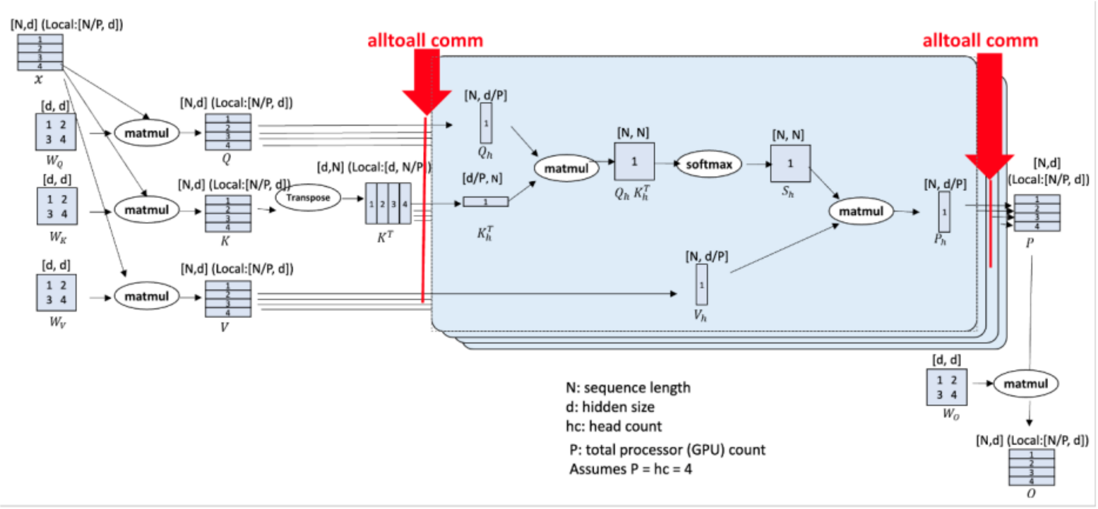



```
Variable Sequence Length Training for Long-Context Large Language
Model, 2023
```

Long-Context Training Systems

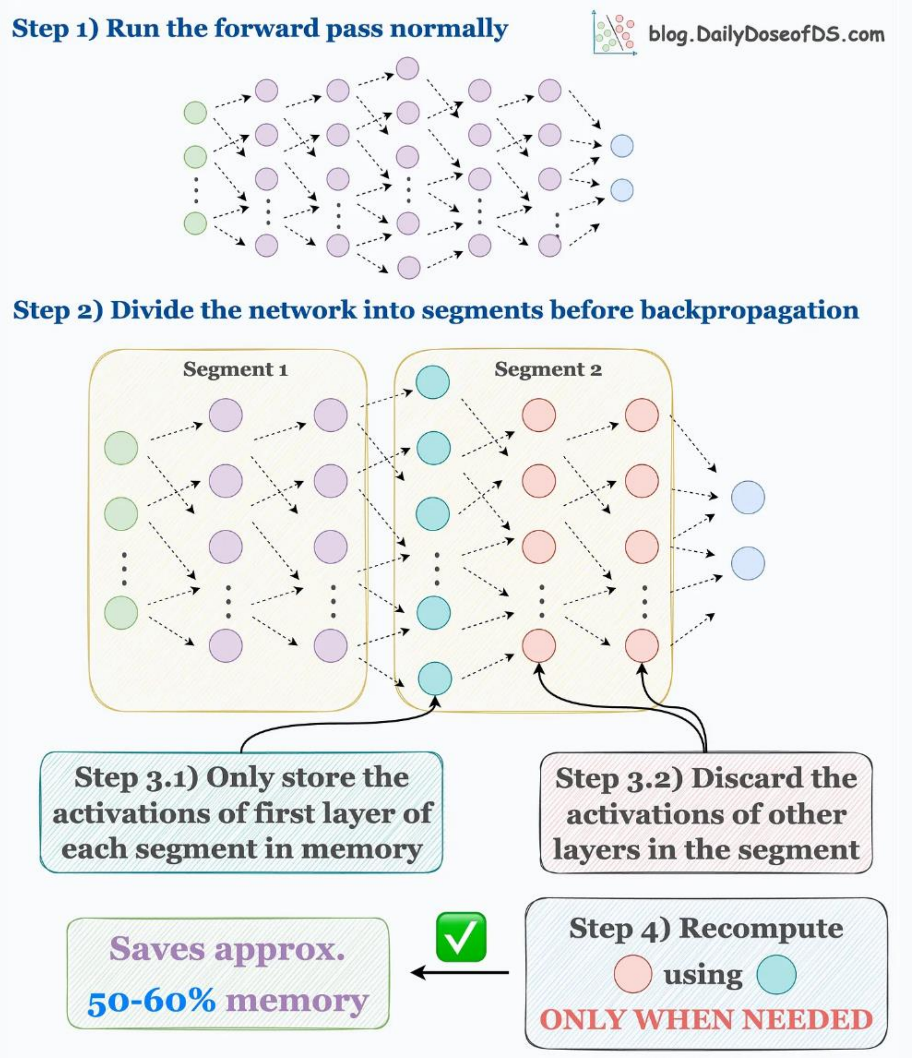
DeepSpeed Ulysses: System Optimizations for Enabling Training of Extreme Long Sequence Transformer Models [PODC 2024]

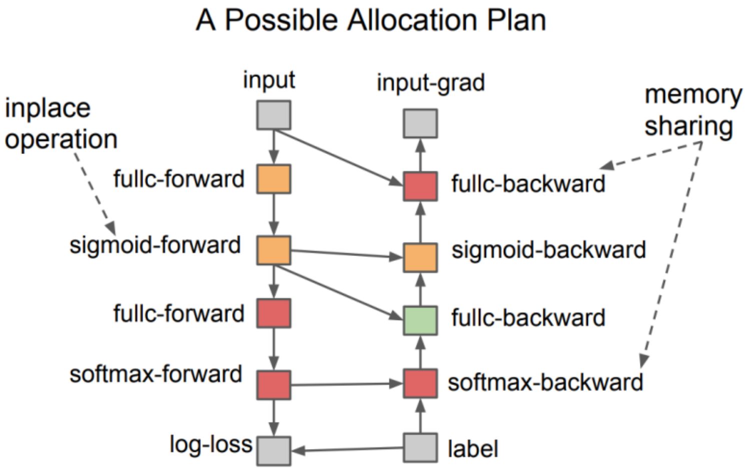




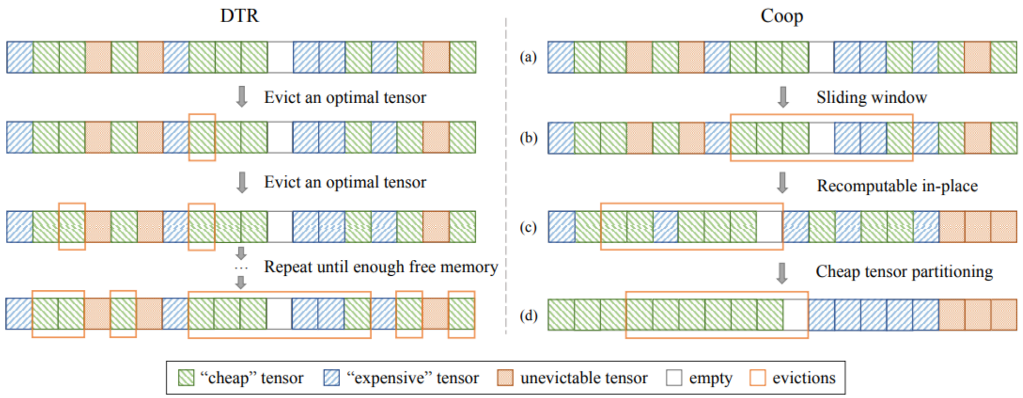
Reducing Activation Recomputation in Large Transformer Models [Arxiv 2022]

Gradient Checkpointing/Rematerialization



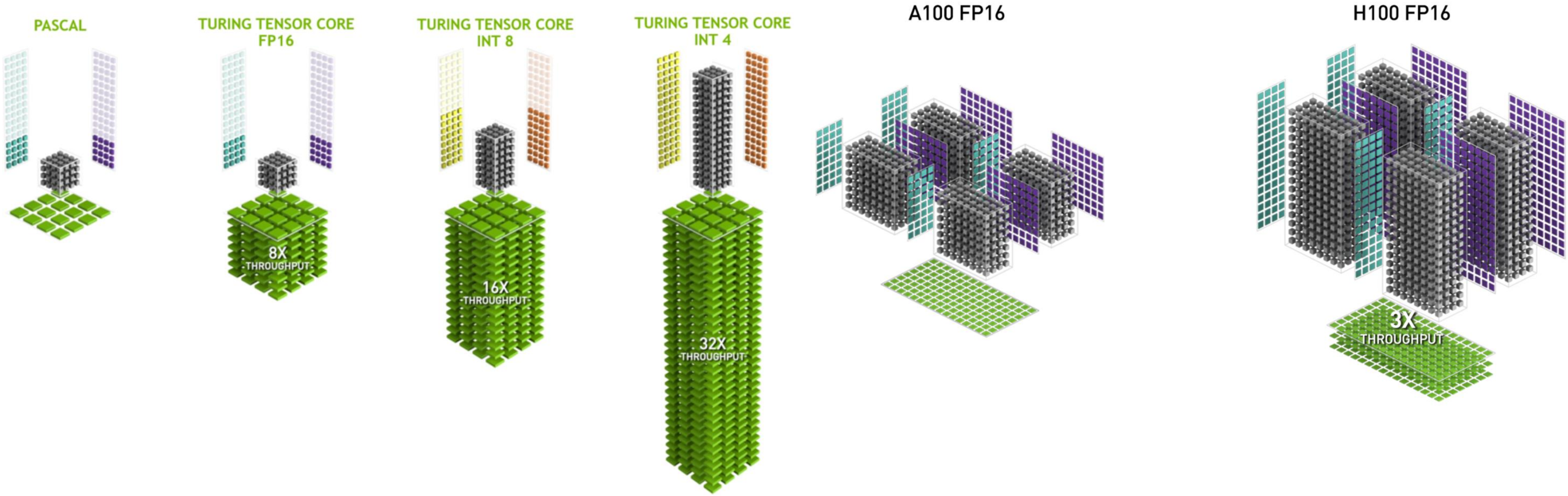


Training Deep Nets with Sublinear Memory Cost, 2016



Coop: Memory is not a Commodity, 2023

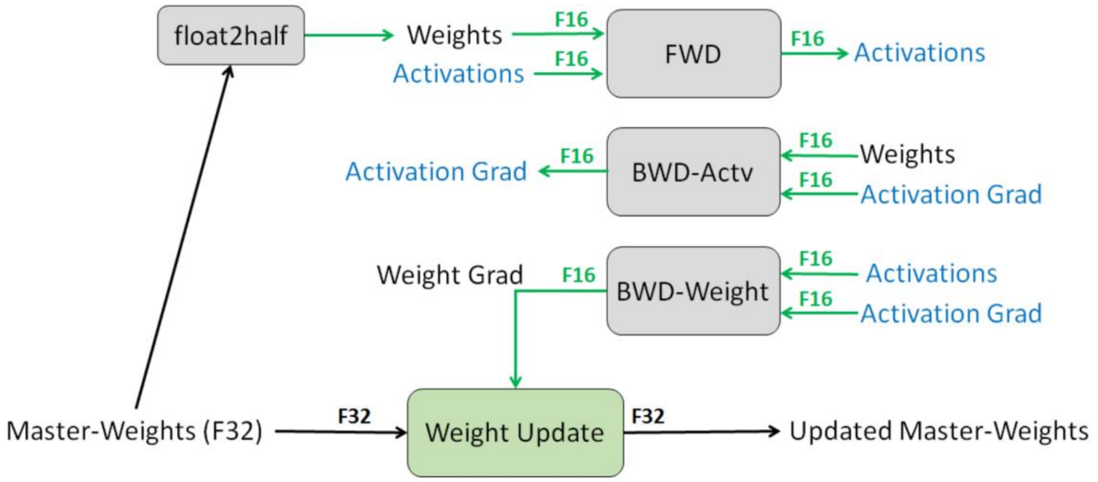
New Hardware Support



Introducing the NVIDIA H100 Tensor Core GPU [2022]

Mixed Precision Training

FP8-LM: Training FP8 Large Language Models [Arxiv, 2023]



Mixed Precision Training [ICLR, 2018]

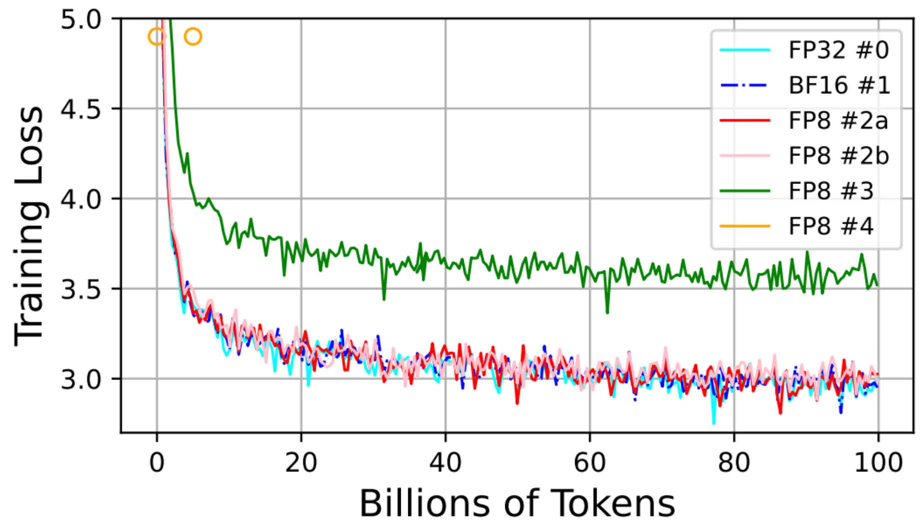
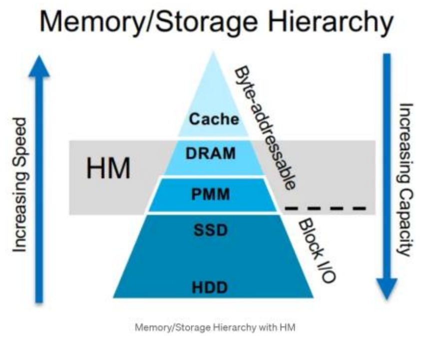
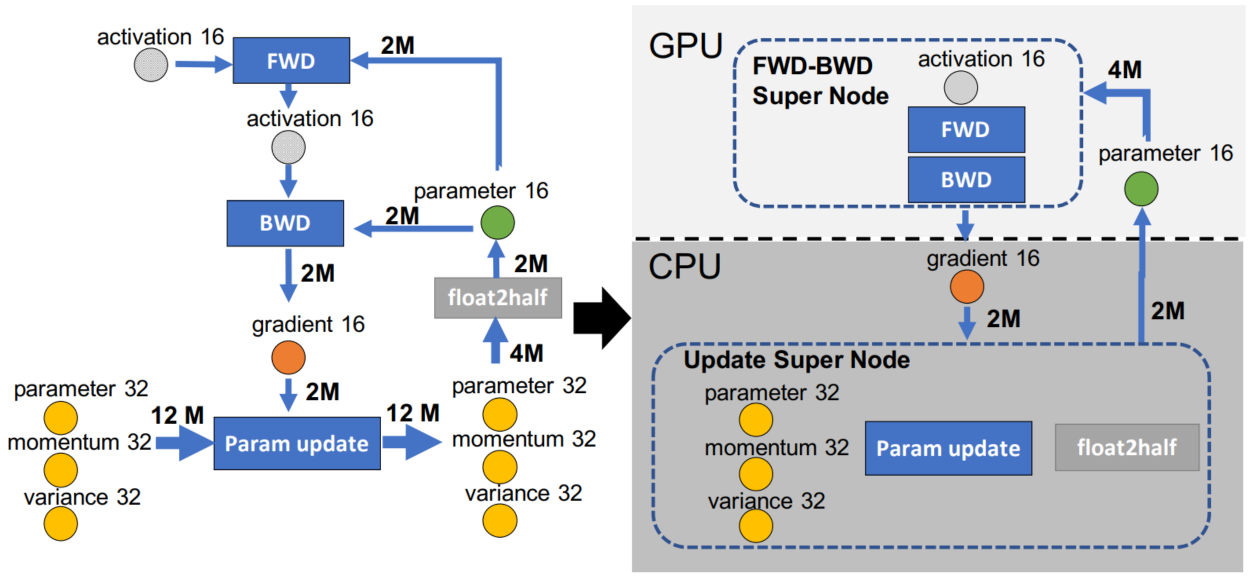


Figure 8: Training losses of GPT-125M models with the settings presented in Tab. 6. The loss curve for FP8 #4 has diverged.

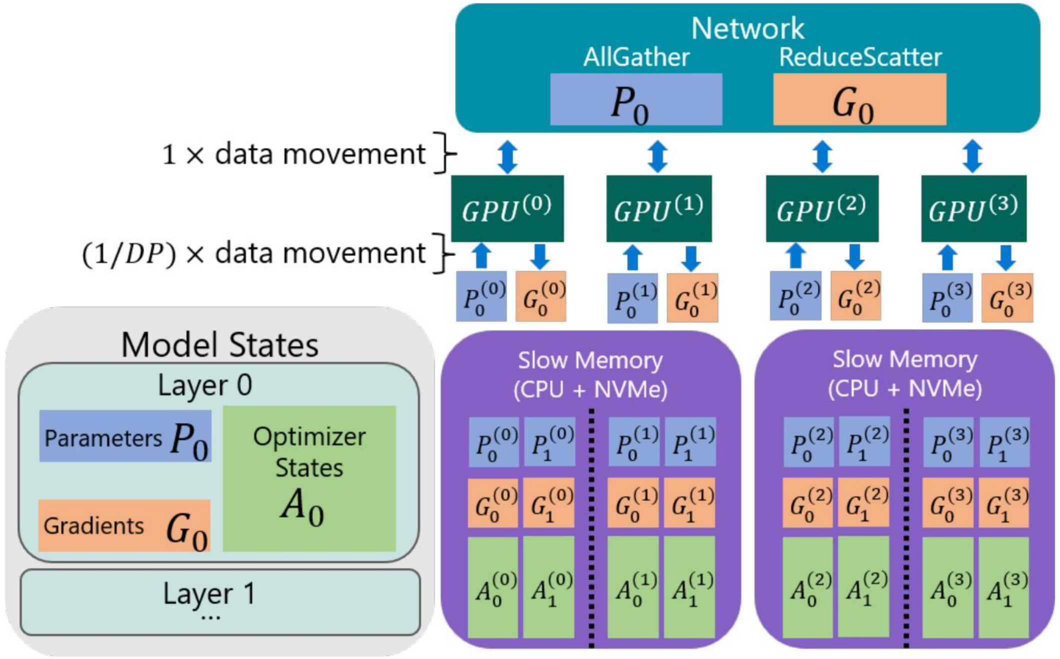
Heterogeneous Memory



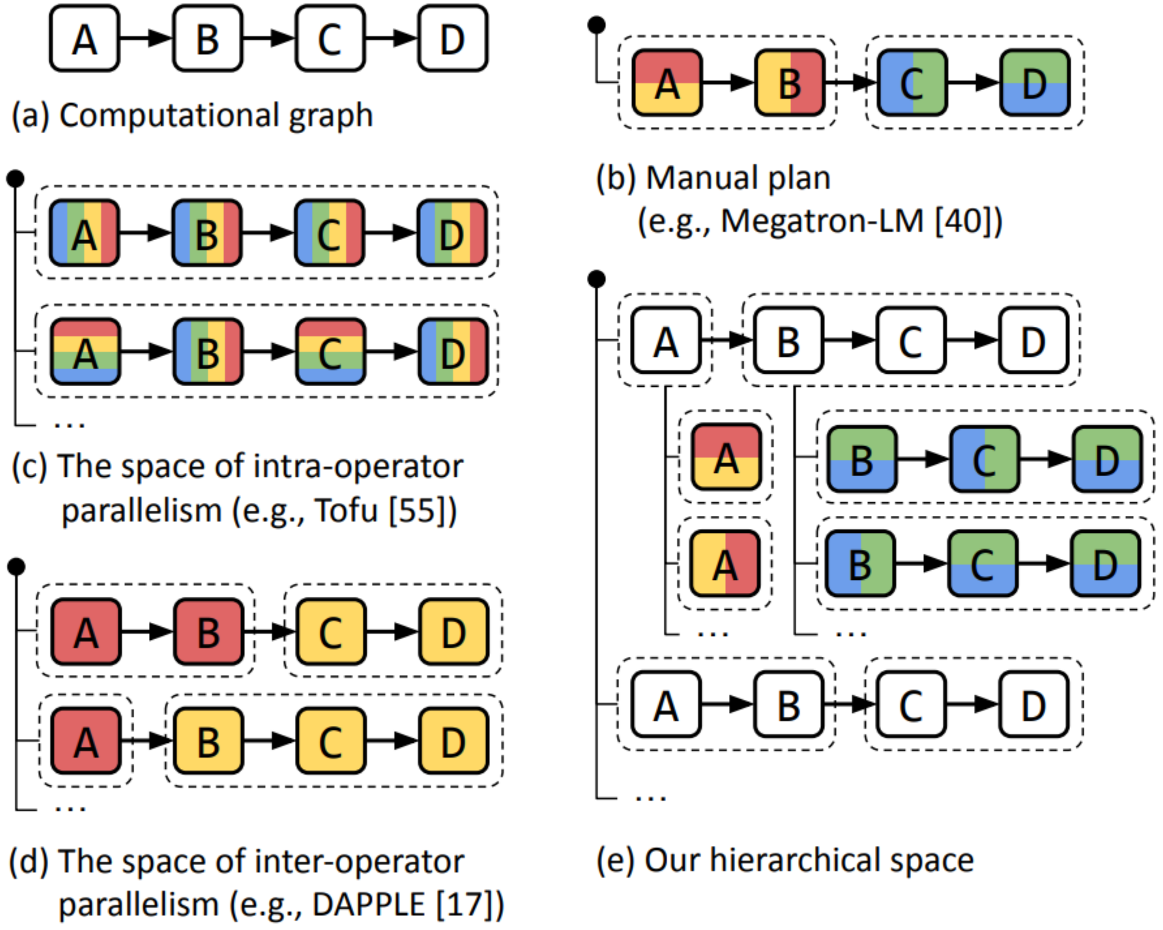


ZeRO-Offload: Democratizing Billion-Scale Model Training [ATC, 2021]

ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning [SC, 2021]

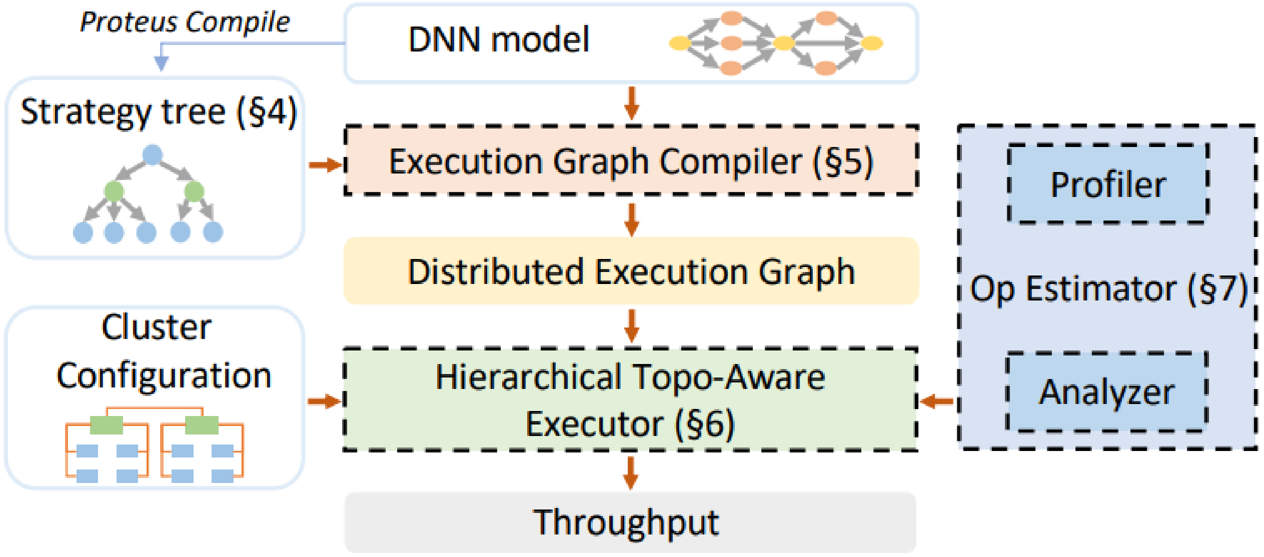


Auto-Parallelism for Easy-to-Use



Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning [OSDI, 2022]

Proteus: Simulating the Performance of Distributed DNN Training [Arxiv, 2023]



QA