CS 598

AI Efficiency: Systems and Algorithms

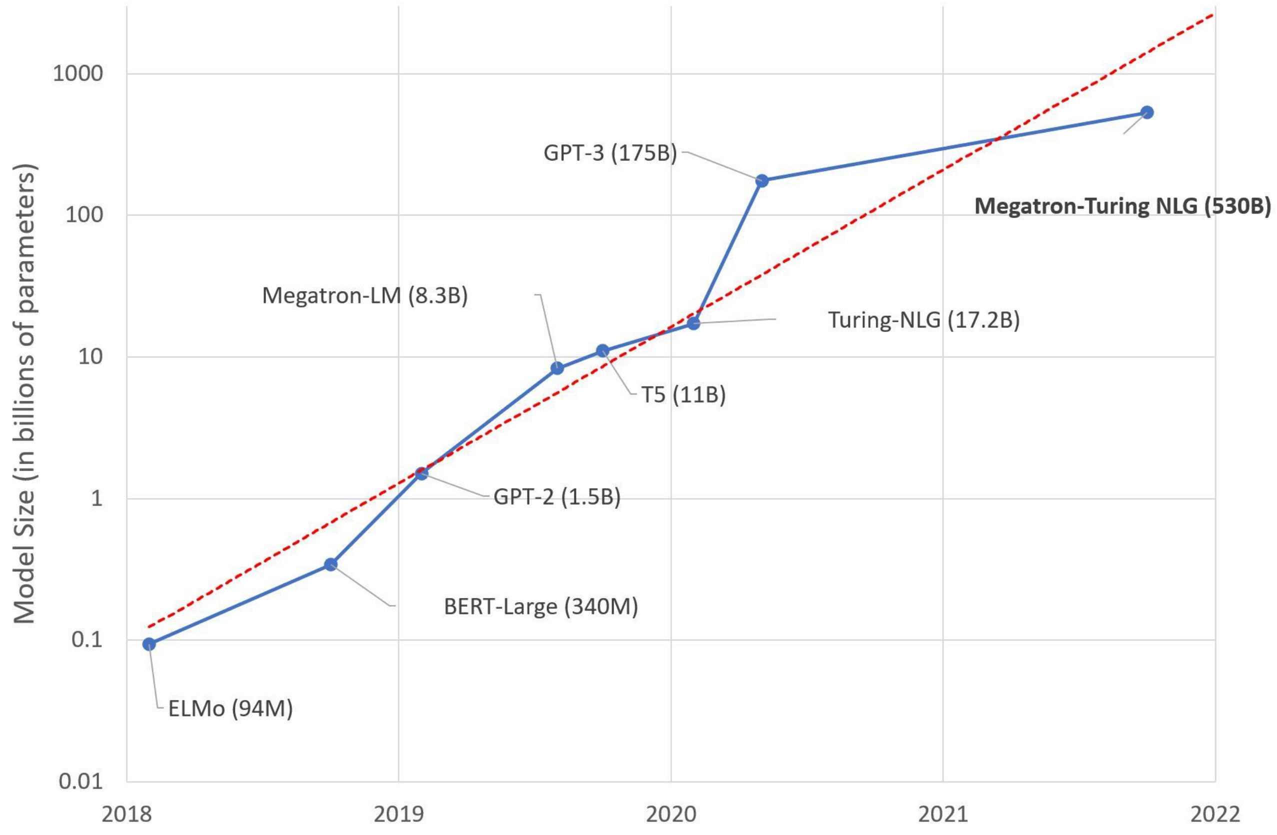
Minjia Zhang

Computer Science Department
University of Illinois at Urbana-Champaign

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

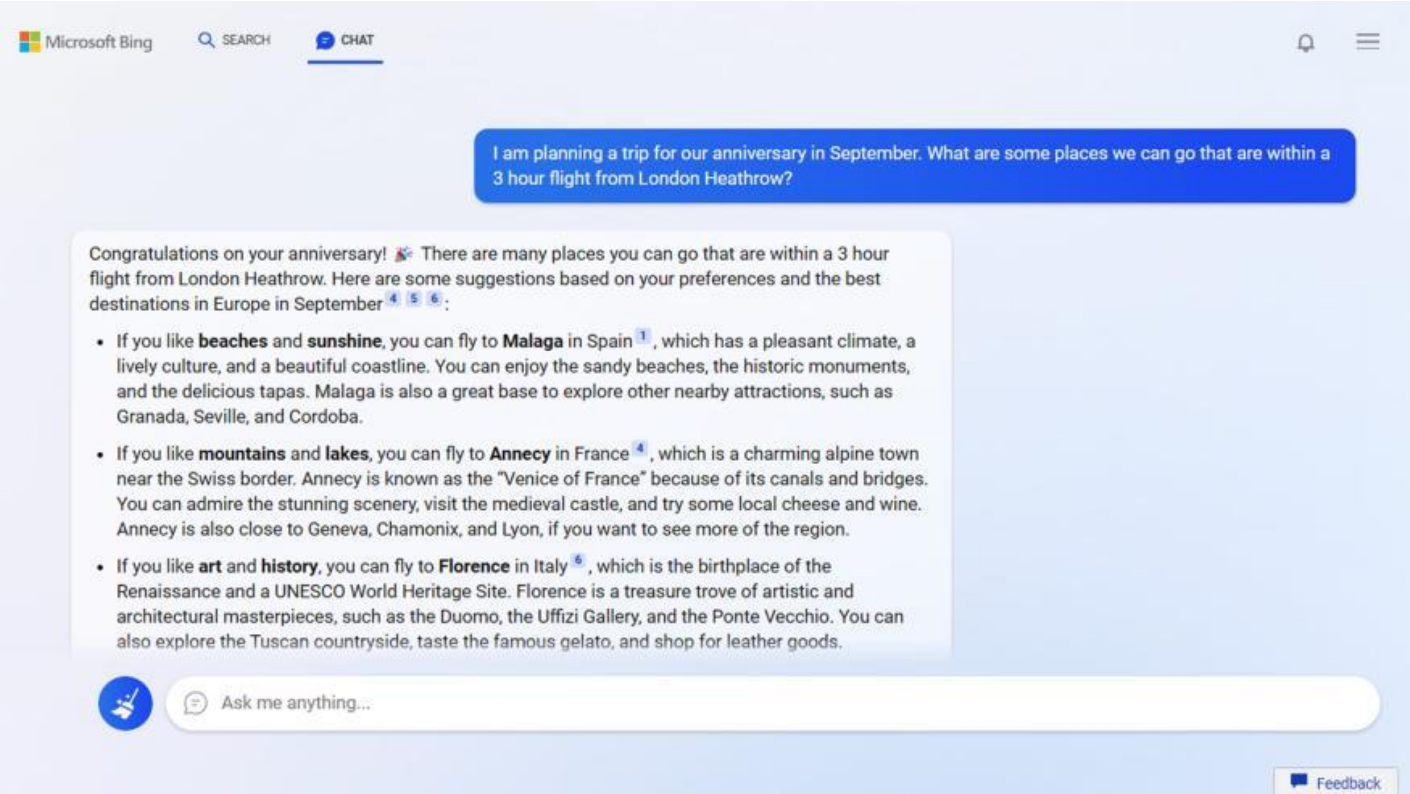
More compute-efficient to train larger models than smaller ones to same accuracy

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]

Dialogue/New Search



GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models

Tyna Eloundou[1], Sam Manning[1,2], Pamela Mishkin*[1], and Daniel Rock[3]

[1]OpenAI
[2]OpenResearch
[3]University of Pennsylvania

March 21, 2023

Abstract

We investigate the potential implications of Generative Pre-trained Transformer (GPT) models and related technologies on the U.S. labor market. Using a new rubric, we assess occupations based on their correspondence with GPT capabilities, incorporating both human expertise and classifications from GPT-4. Our findings indicate that approximately 80% of the U.S. workforce could have at least 10% of their work tasks affected by the introduction of GPTs, while around 19% of workers may see at least 50% of their tasks impacted. The influence spans all wage levels, with higher-income jobs potentially facing greater exposure. Notably, the impact is not limited to industries with higher recent productivity growth. We conclude that Generative Pre-trained Transformers exhibit characteristics of general-purpose technologies (GPTs), suggesting that as these models could have notable economic, social, and policy implications.

[ChatGPT: Optimizing Language Models for Dialogue]

Image Generation from Text

TEXT DESCRIPTION

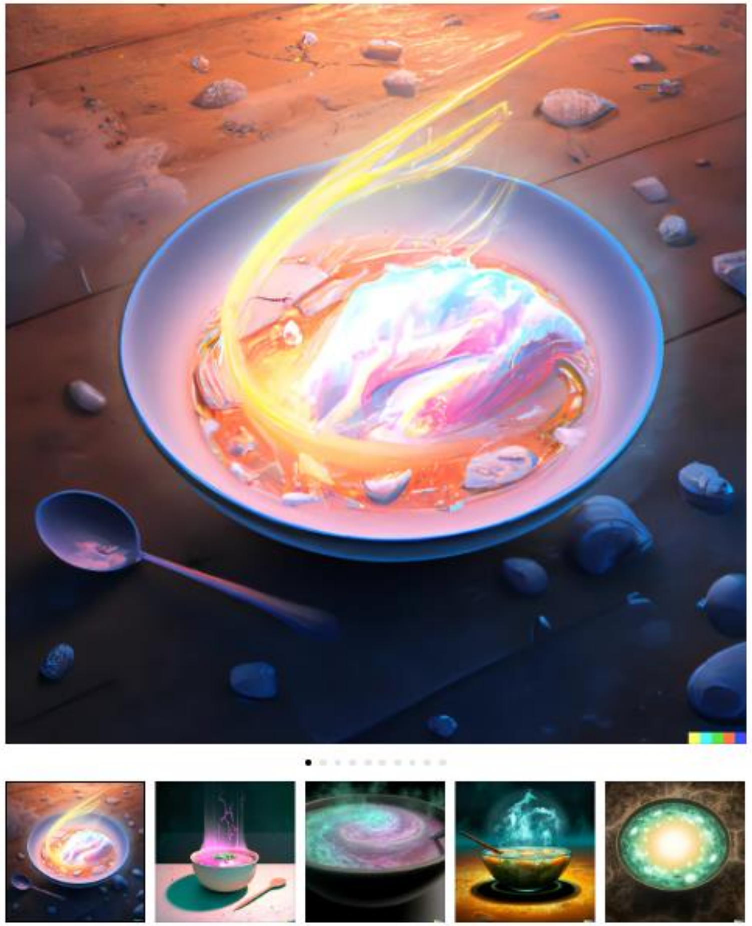
An astronaut Teddy bears A bowl of soup

that is a portal to another dimension that looks like a monster as a planet in the universe

as digital art in the style of Basquiat drawn on a cave wall

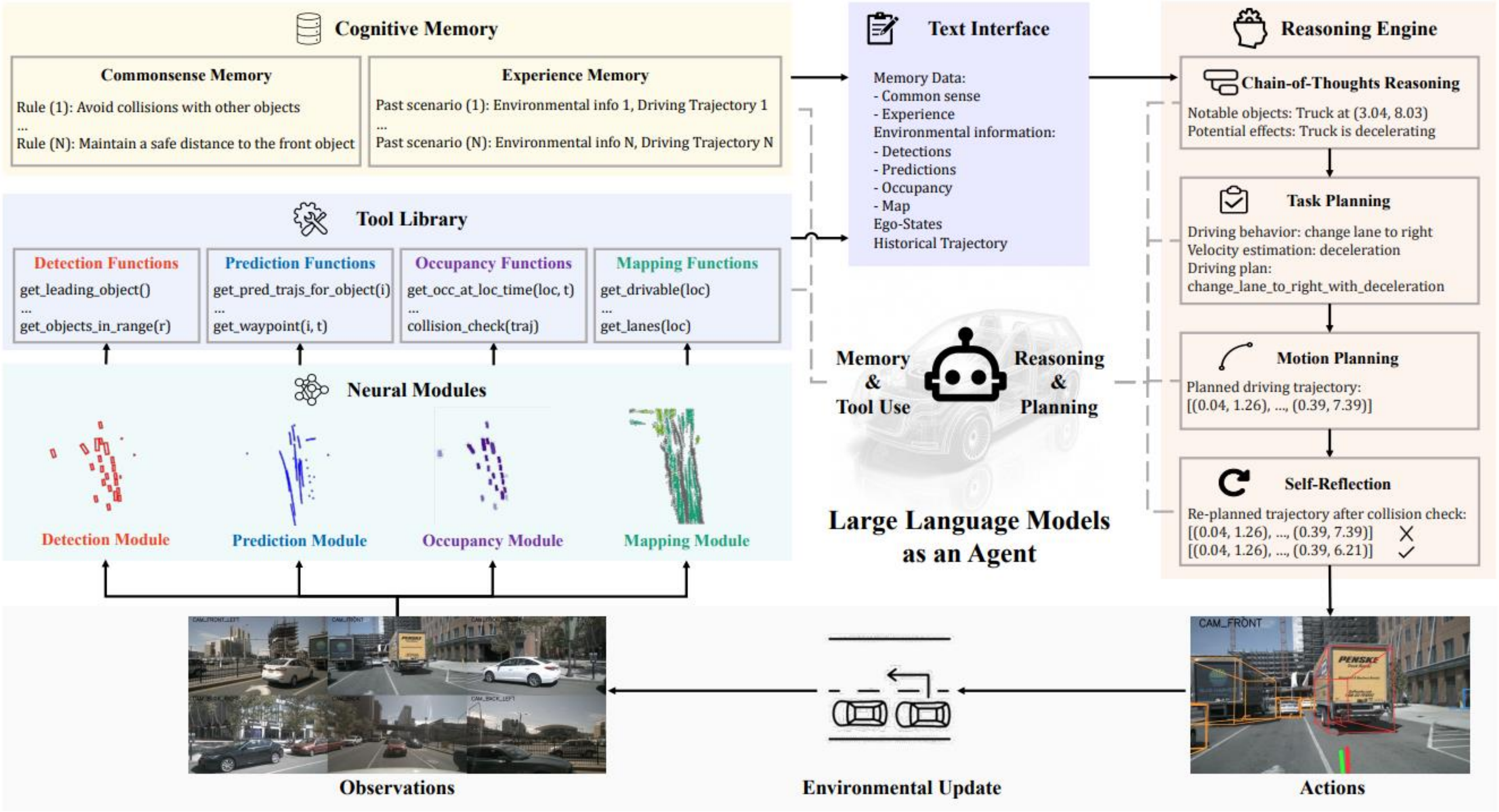
→

DALL·E 2



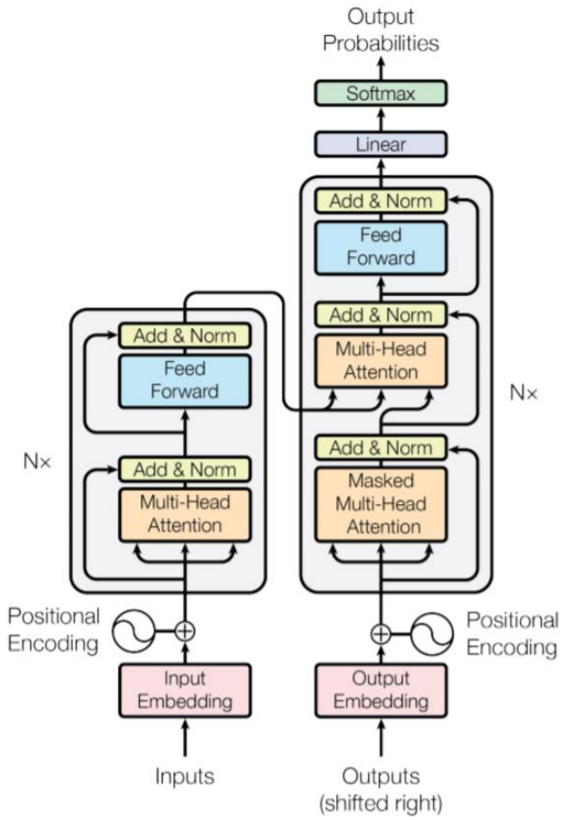
[DALL·E: Creating Images from Text - OpenAI]

Autonomous Driving

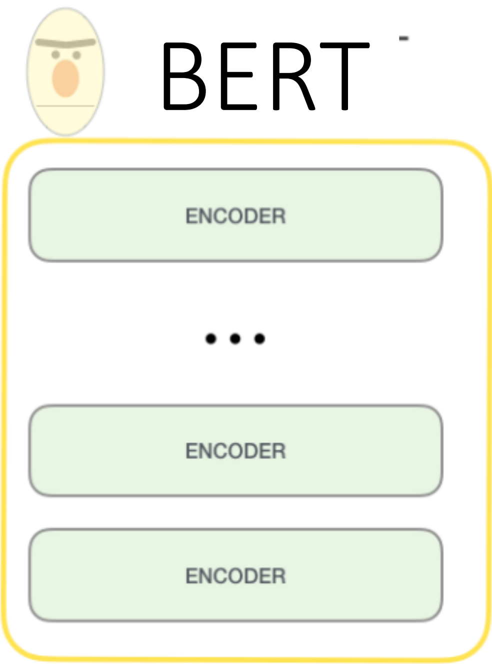


[A Language Agent for Autonomous Driving]

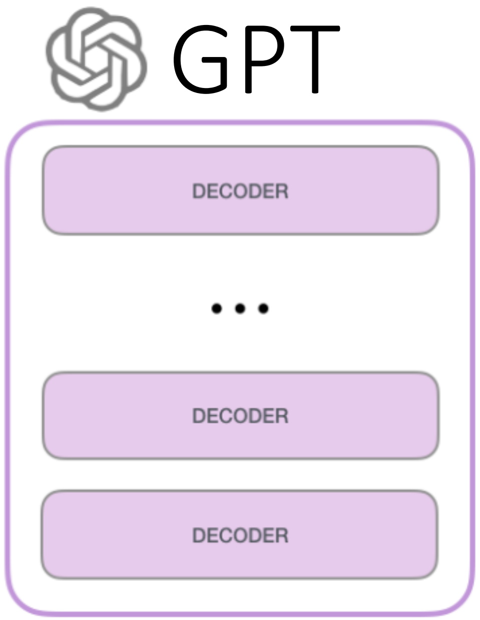
Transformers for Language Modeling



[1]



[2]



[3]

[1] Vaswani et al. “Attention Is All You Need”, https://arxiv.org/abs/1706.03762, 2018
[2]Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, 2019, https://arxiv.org/abs/1810.04805
[3] Brown et al. “Language Models are Few-Shot Learners”, 2020, https://arxiv.org/abs/2005.14165

AI Efficiency Challenges

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

Research Focus (3Es)

Efficiency

Efficient use of hardware for low latency, high scalability and throughput

- ❑ **Training efficiency**

- High performance and cost-efficient training [ATC'21]
  (Training **10x larger** model with **same GPUs**, **adopted by major DL frameworks**)
- Graph neural networks [ASPLOS'23]
- Training w. spot instances [NSDI'23, SOSP'24]

- ❑ **Inference efficiency**

- Recurrent neural networks [ATC'18]
  (**10x faster** latency and **hundreds of millions of dollar saving**, Microsoft 2018 **top-3 "Cool Tech" showcase**)
- Transformers [SC'22]

Research Focus (3Es)

Efficiency

Efficient use of hardware for low latency, high scalability and throughput

❑ Training efficiency

- High performance and cost-efficient training [ATC'21]
  (Training **10x larger** model with **same GPUs**, **adopted by major DL frameworks**)
- Graph neural networks [ASPLOS'23]
- Training w. spot instances [NSDI'23, SOSP'24]

❑ Inference efficiency

- Recurrent neural networks [ATC'18]
  (**10x faster** latency and **hundreds of millions of dollar saving**, Microsoft 2018 **top-3 "Cool Tech" showcase**)
- Transformers [SC'22]

Effectiveness

High accuracy and fast convergence, lowering cost

❑ Model compression

- Extreme compression [NeurIPS'22 **Oral**]
  (**50x smaller** model size, similar accuracy)
- Zero-cost quant. [NeurIPS'22 **spotlight**]
- 1-bit communication [ICLR'23]

❑ Data efficiency

- Curriculum learning [NeurIPS'22, **spotlight**]
  (Retaining 99% accuracy with **10x less data**)
- Adversarial learning [AAAI'22]

❑ Efficient architecture

- Mixture-of-Experts [ICML'22]

Research Focus (3Es)

Efficiency

Efficient use of hardware for low latency, high scalability and throughput

- ❑ **Training efficiency**
  - High performance and cost-efficient training [ATC'21] (Training **10x larger** model with **same GPUs**, **adopted by major DL frameworks**)
  - Graph neural networks [ASPLOS'23]
  - Training w. spot instances [NSDI'23, SOSP'24]
- ❑ **Inference efficiency**
  - Recurrent neural networks [ATC'18] (**10x faster** latency and **hundreds of millions of dollar saving**, Microsoft 2018 **top-3 "Cool Tech" showcase**)
  - Transformers [SC'22]

Effectiveness

High accuracy and fast convergence, lowering cost

- ❑ **Model compression**
  - Extreme compression [NeurIPS'22 **Oral**] (**50x smaller** model size, similar accuracy)
  - Zero-cost quant. [NeurIPS'22 **spotlight**]
  - 1-bit communication [ICLR'23]
- ❑ **Data efficiency**
  - Curriculum learning [NeurIPS'22, **spotlight**] (Retaining 99% accuracy with **10x less data**)
  - Adversarial learning [AAAI'22]
- ❑ **Efficient architecture**
  - Mixture-of-Experts [ICML'22]

Easy-to-Use

Improve development productivity of model scientists

- ❑ **DL Compilation**
  - Hardware heterogeneity [IPDPS'21]
- ❑ **Auto-Tuner**
  - Adaptive tuning [NeurIPS'20] (**3.9x faster** optimization speed)
  - Multi-task tuning [ICLR'21]

Research Focus (3Es)

Efficiency

Efficient use of hardware for low latency, high scalability and throughput

- ❑ **Training efficiency**
  - High performance and cost-efficient training [ATC'21] (Training **10x larger** model with **same GPUs**, **adopted by major DL frameworks**)
  - Graph neural networks [ASPLOS'23]
  - Training w. spot instances [NSDI'23, SOSP'24]
- ❑ **Inference efficiency**
  - Recurrent neural networks [ATC'18] (**10x faster** latency and **hundreds of millions of dollar saving**, Microsoft 2018 **top-3 "Cool Tech" showcase**)
  - Transformers [SC'22]

Effectiveness

High accuracy and fast convergence, lowering cost

- ❑ **Model compression**
  - Extreme compression [NeurIPS'22 **Oral**] (**50x smaller** model size, similar accuracy)
  - Zero-cost quant. [NeurIPS'22 **spotlight**]
  - 1-bit communication [ICLR'23]
- ❑ **Data efficiency**
  - Curriculum learning [NeurIPS'22, **spotlight**] (Retaining 99% accuracy with **10x less data**)
  - Adversarial learning [AAAI'22]
- ❑ **Efficient architecture**
  - Mixture-of-Experts [ICML'22]

Easy-to-Use

Improve development productivity of model scientists

- ❑ **DL Compilation**
  - Hardware heterogeneity [IPDPS'21]
- ❑ **Auto-Tuner**
  - Adaptive tuning [NeurIPS'20] (**3.9x faster** optimization speed)
  - Multi-task tuning [ICLR'21]

DeepSpeed tvm

Industry products: Bing, Ads, Azure, Office

Open-source software

Training Efficiency: Breaking the Memory Wall

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
- η > 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 -= $\boxed{\Delta w}$ // 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*

[1] Kingma and Ba, "Adam: A Method for Stochastic Optimization", 2014, https://arxiv.org/abs/1412.6980

Parallel/Distributed Gradient Descent

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

Can we parallelize it?

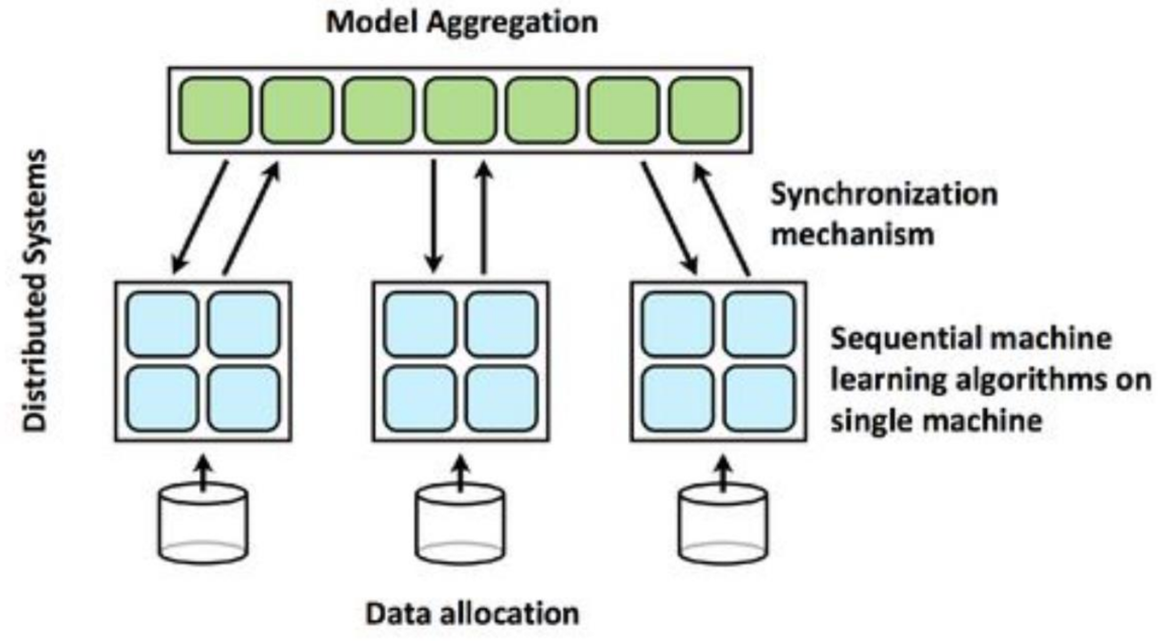
For t = 1 to T

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

w -= Δw // apply update

End

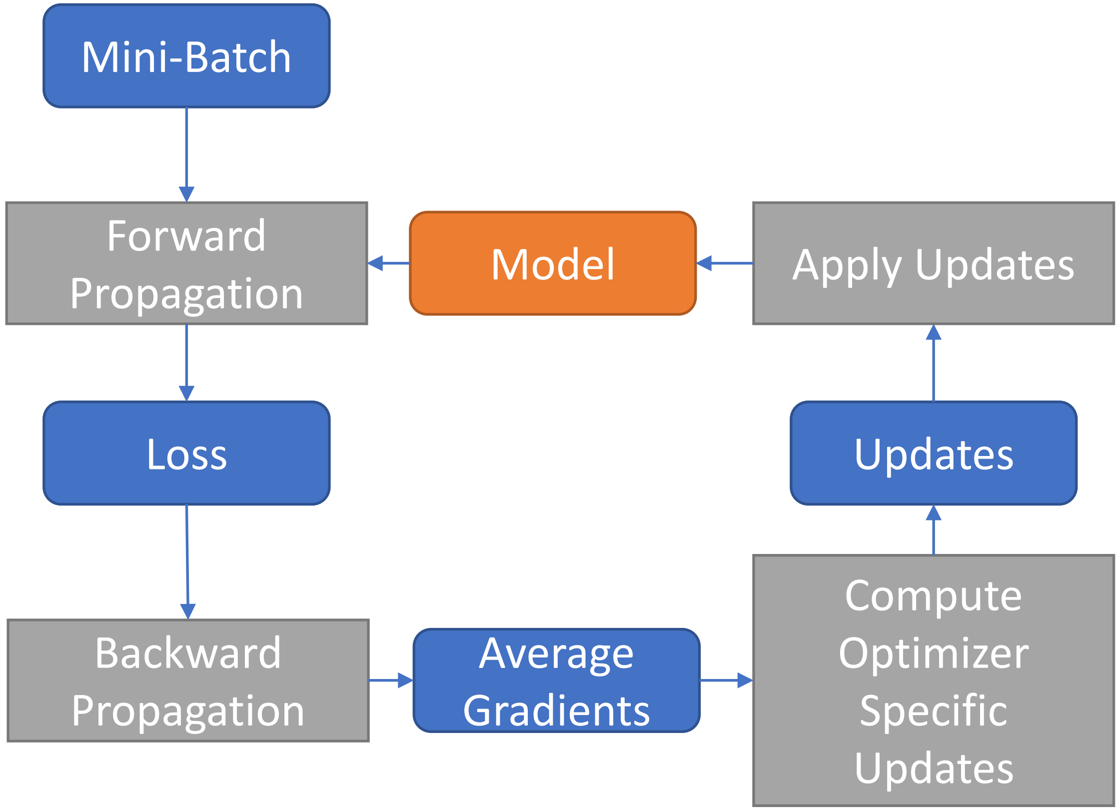
Data Parallelism (DP)



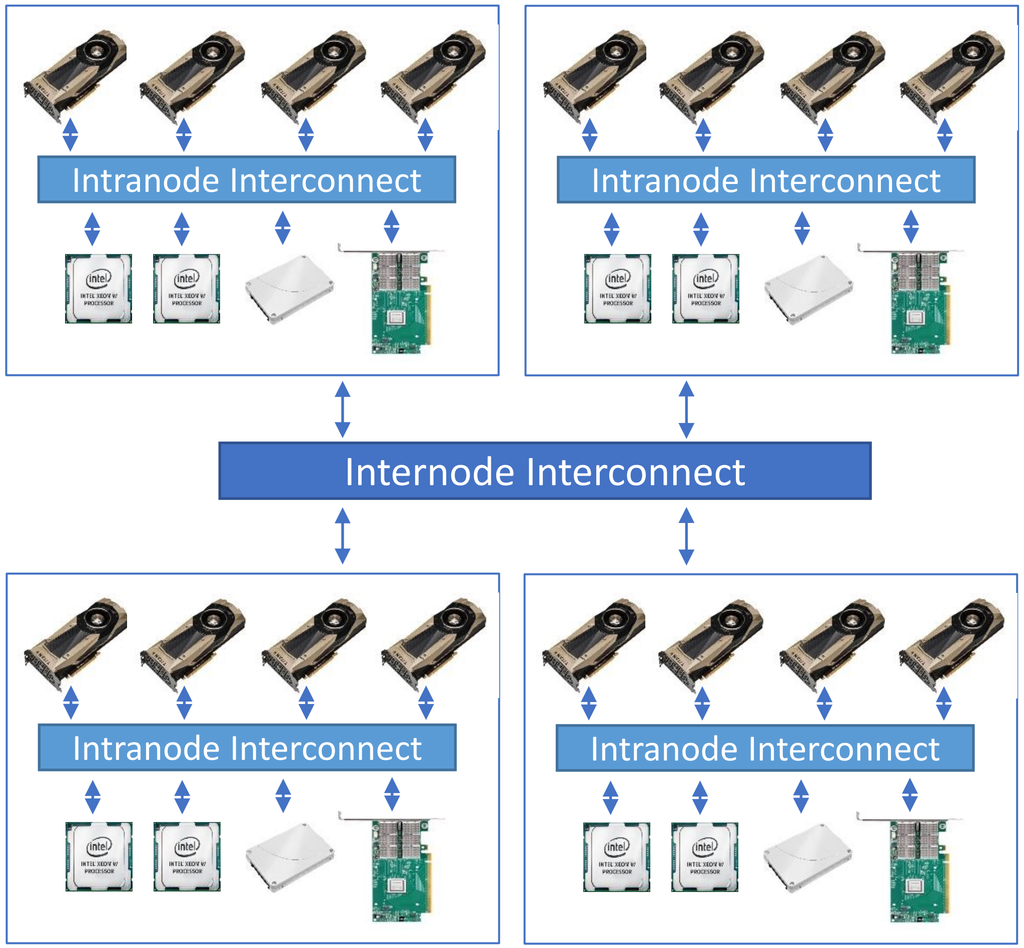
1. Partition the training data
2. Parallel training on different machines
3. Synchronize the local updates
4. Refresh local model with new parameters, then go to 2

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

Distributed Data Parallel Training in GPU Clusters



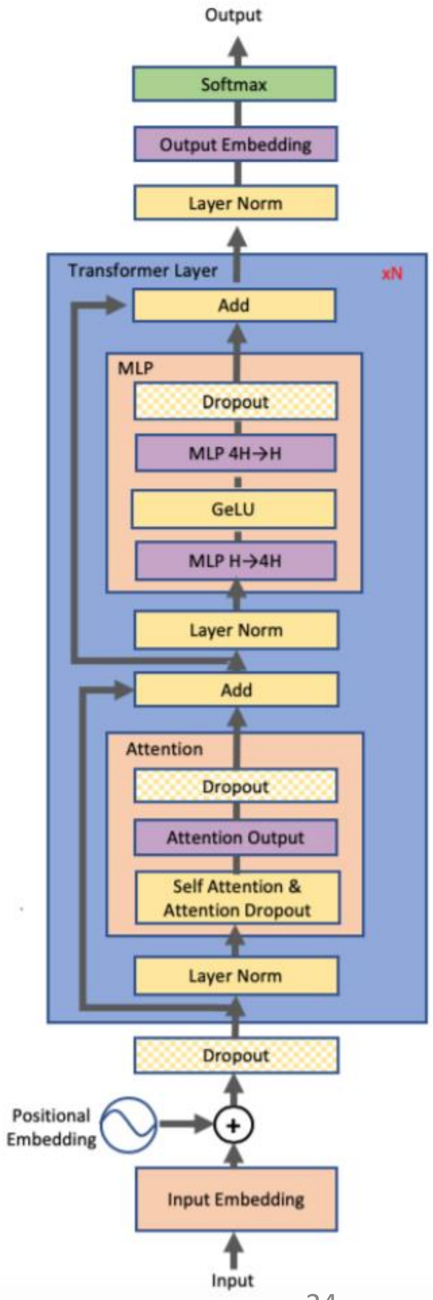
Data Parallel Training Loop



Distributed GPU Cluster

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**

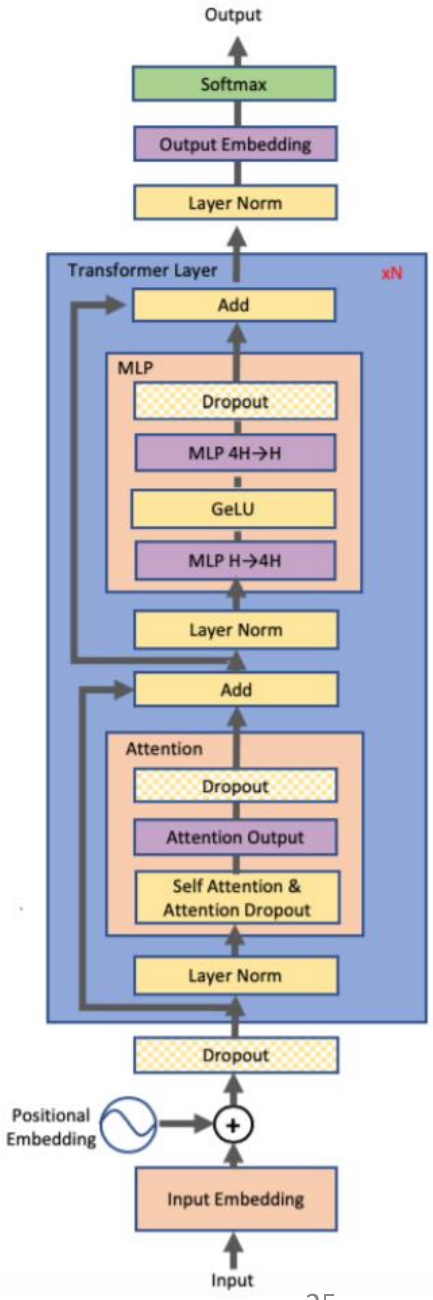


Large Model Training Challenges

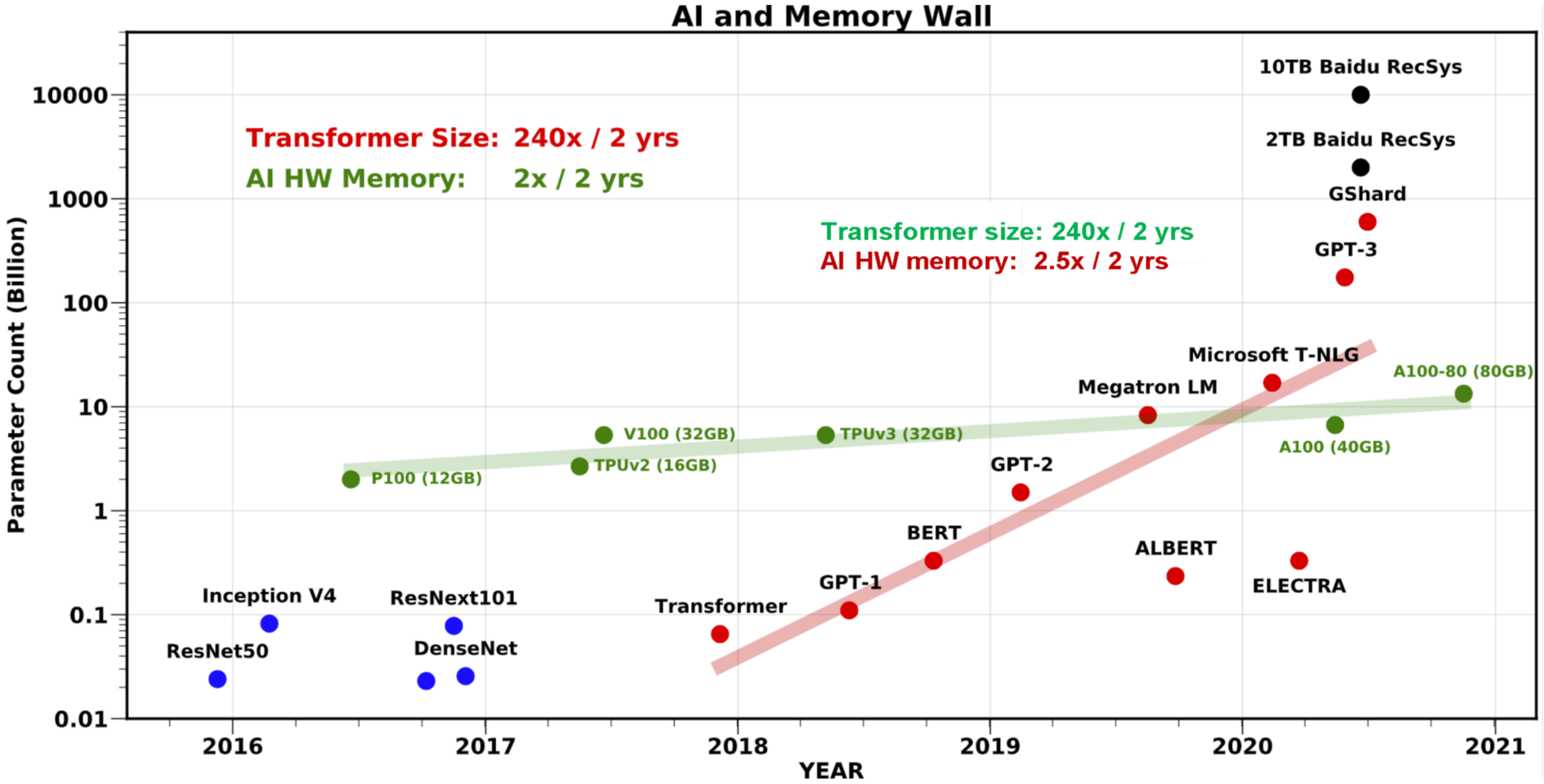
	Bert-Large	GPT-2	Turing 17.2 NLG	GPT-3
Parameters	0.32B	1.5B	17.2B	175B
Layers	24	48	78	96
Hidden Dimension	1024	1600	4256	12288
Relative Computation	1x	4.7x	**54x**	**547x**
Memory Footprint	5.12GB	24GB	**275GB**	**2800GB**

Out of Memory

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



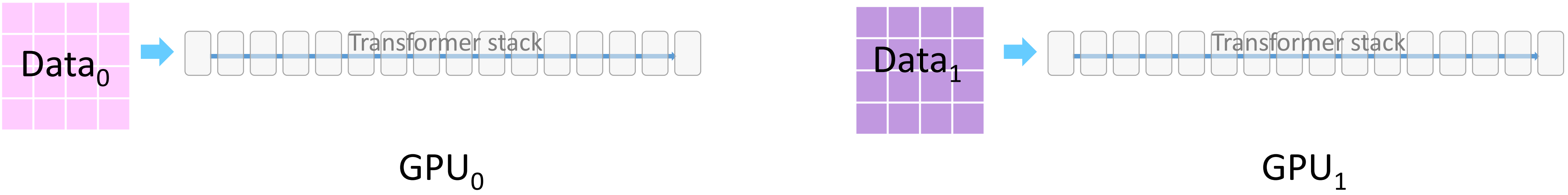
DNN Training Hit the Memory Wall



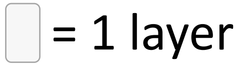
*AI and Memory Wall. (This blogpost has been written in… | by Amir Gholami | riselab | Medium

How to Break the Memory Wall?

Understanding Memory Consumption

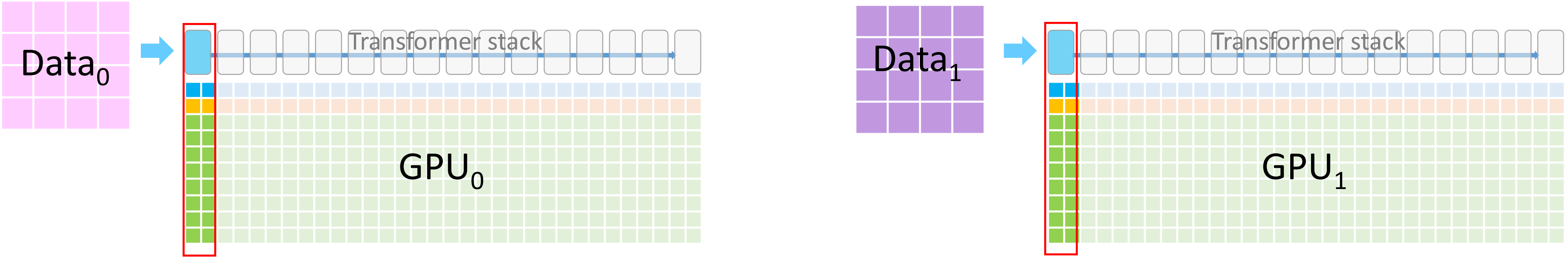


A 16-layer transformer model



*Mixed Precision Training (ICLR '18) with Adam Optimizer

Understanding Memory Consumption



Each cell represents GPU memory used by its corresponding transformer layer

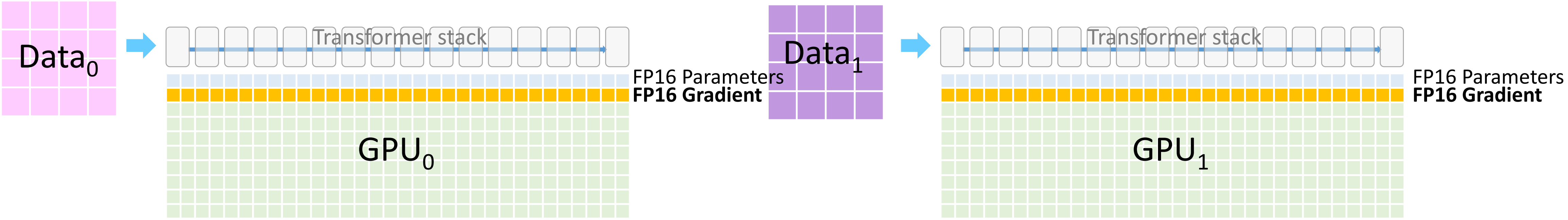
*Mixed Precision Training (ICLR ‘18) with Adam Optimizer

Understanding Memory Consumption

- FP16 parameter

*Mixed Precision Training (ICLR ‘18) with Adam Optimizer

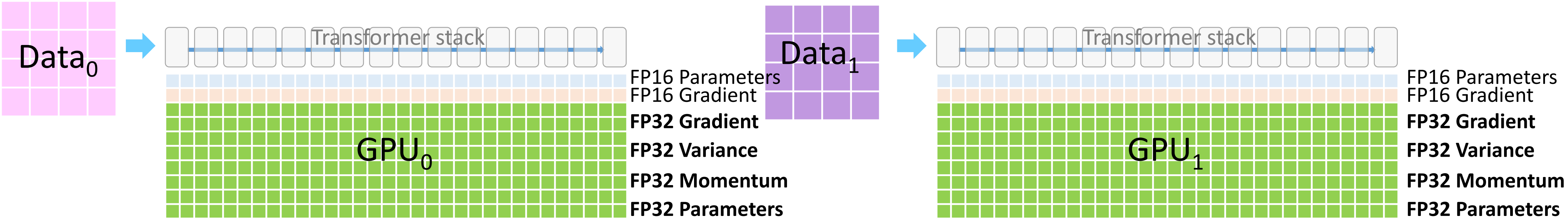
Understanding Memory Consumption



- FP16 parameter
- FP16 Gradients

*Mixed Precision Training (ICLR '18) with Adam Optimizer

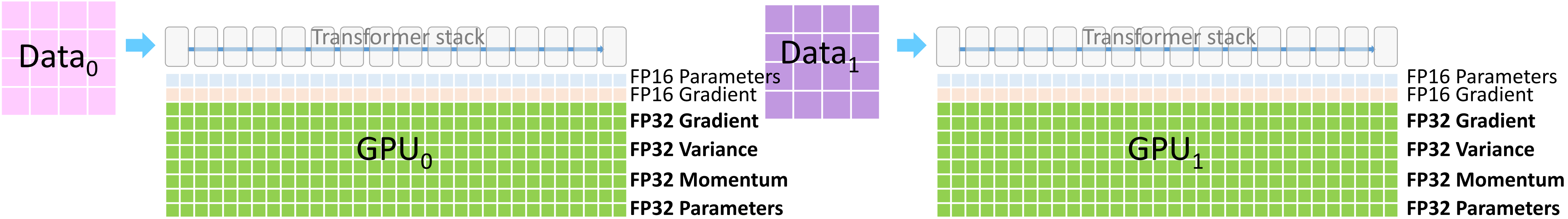
Understanding Memory Consumption



- FP16 parameter
- FP16 Gradients
- FP32 Optimizer States
  - Gradients, Variance, Momentum, Parameters

*Mixed Precision Training (ICLR '18) with Adam Optimizer

Understanding Memory Consumption



- FP16 parameter : **2M bytes**
- FP16 Gradients : **2M bytes**
- FP32 Optimizer States : **16M bytes**
  - Gradients, Variance, Momentum, Parameters

M = number of parameters in the model

Example 1B parameter model -> 20GB/GPU

Memory consumption doesn't include:

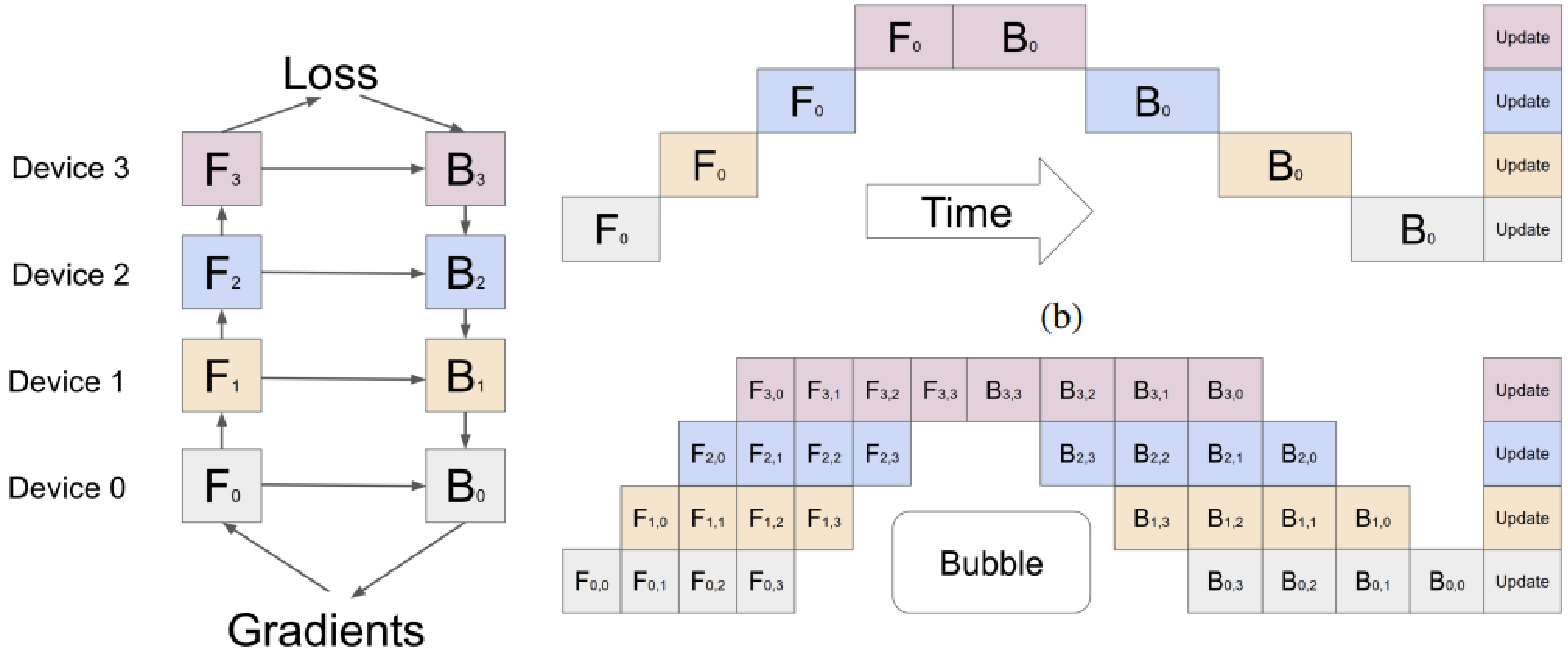
- Input batch + activations

*Mixed Precision Training (ICLR '18) with Adam Optimizer

Distributed Training Strategies

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

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

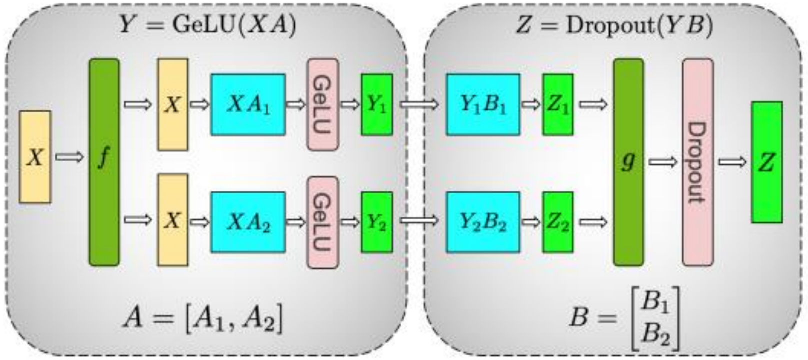
GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism

Tensor Parallelism

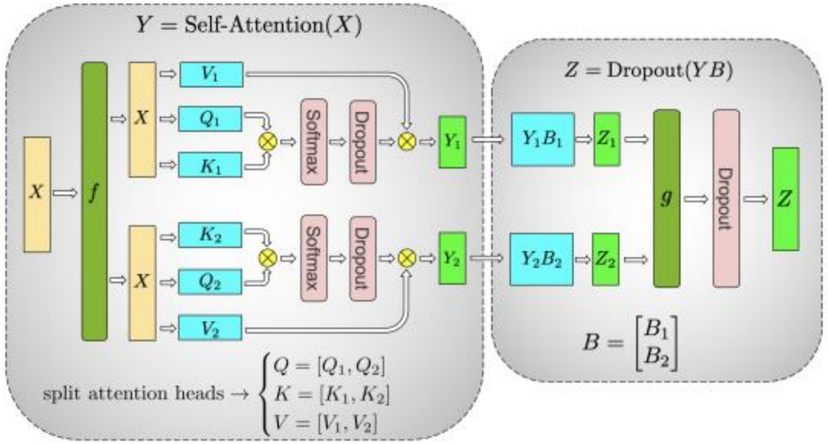
Splice tensors across GPUs

+

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



(a) MLP



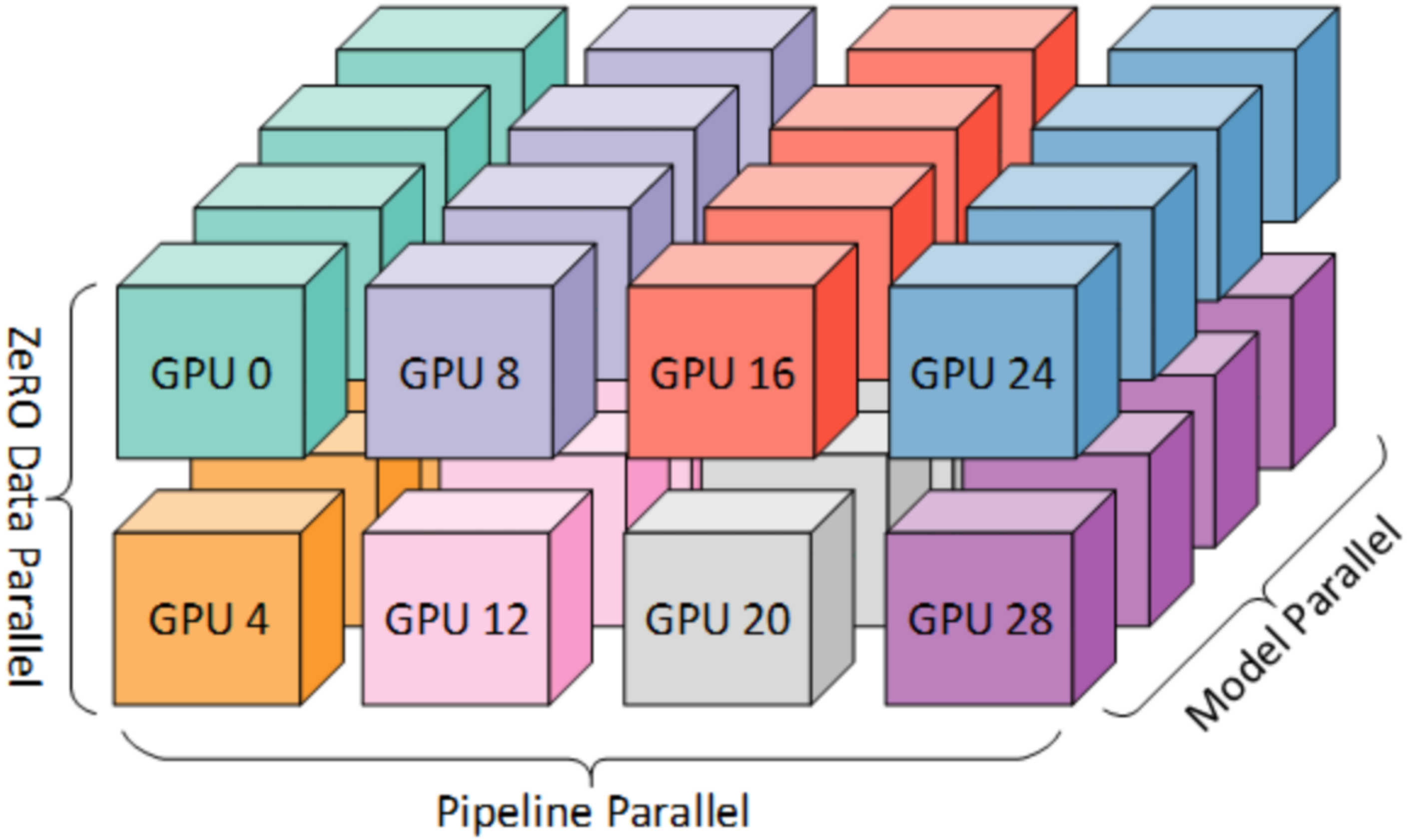
(b) Self-Attention

Supported in:

- DeepSpeed
- Megatron-LM

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

3D Parallelism

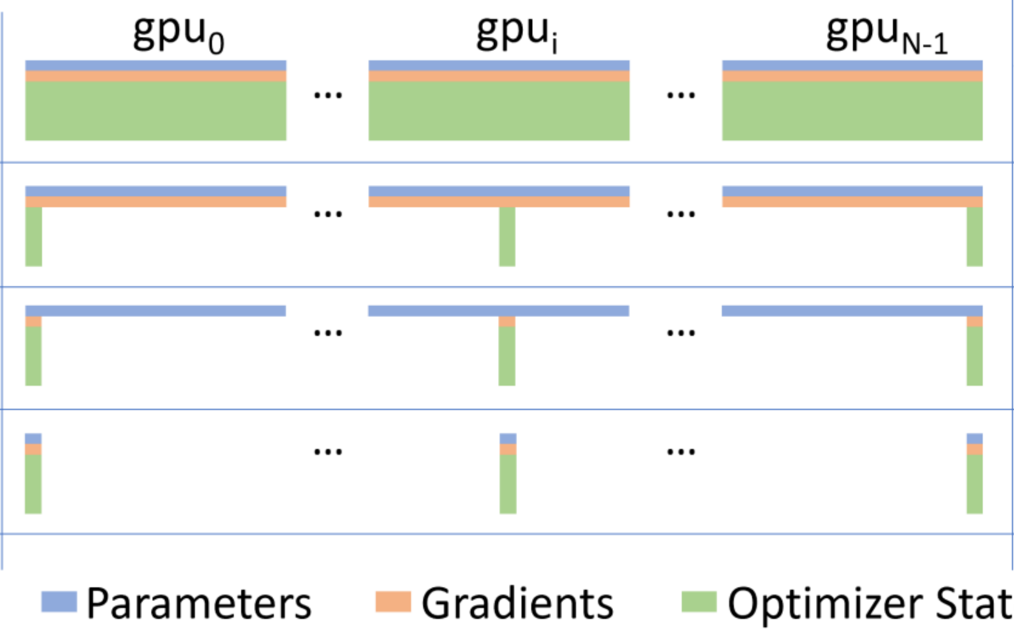


Supported in:
- [DeepSpeed]
- [Megatron-LM]

DeepSpeed-extreme-scale-model-training-for-everyone

ZeRO-Style Data Parallelism

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



	Memory Consumed	K=12 Ψ=7.5B N_d=64
Baseline	$(2+2+K)*\Psi$	120GB
P_{os}	$2\Psi+2\Psi+\frac{K*\Psi}{N_d}$	31.4GB
P_{os+g}	$2\Psi+\frac{(2+K)*\Psi}{N_d}$	16.6GB
P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB

Supported in:
- [DeepSpeed]
- [PyTorch]

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

More Interesting Work on Training

- Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning, OSDI 22
- Sequence Parallelism: Making 4D Parallelism Possible, ACL 2023
- Tutel: An efficient mixture-of-experts implementation for large DNN model training, 2023
- Bamboo: Making Preemptible Instances Resilient for Affordable Training of Large DNNs, NSDI 2023
- …

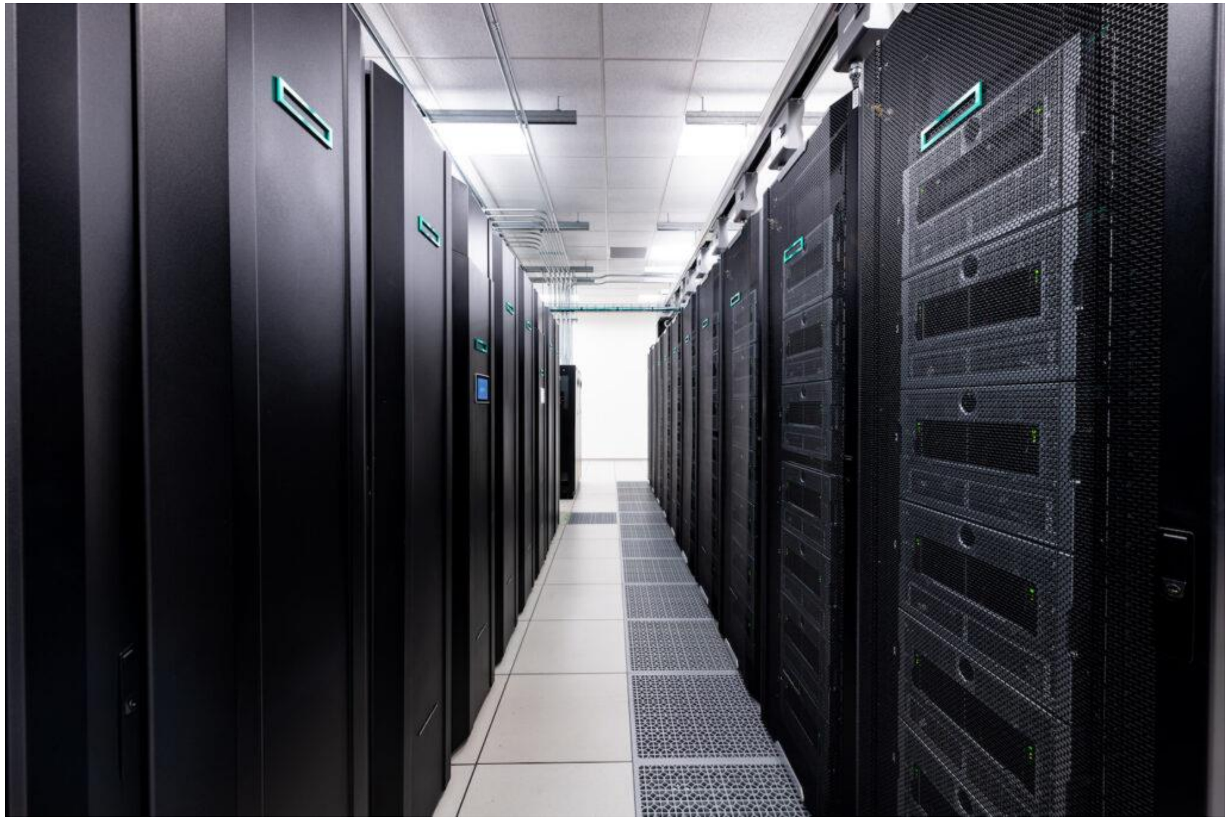
Tentative Schedule

Hardware Resource

Delta

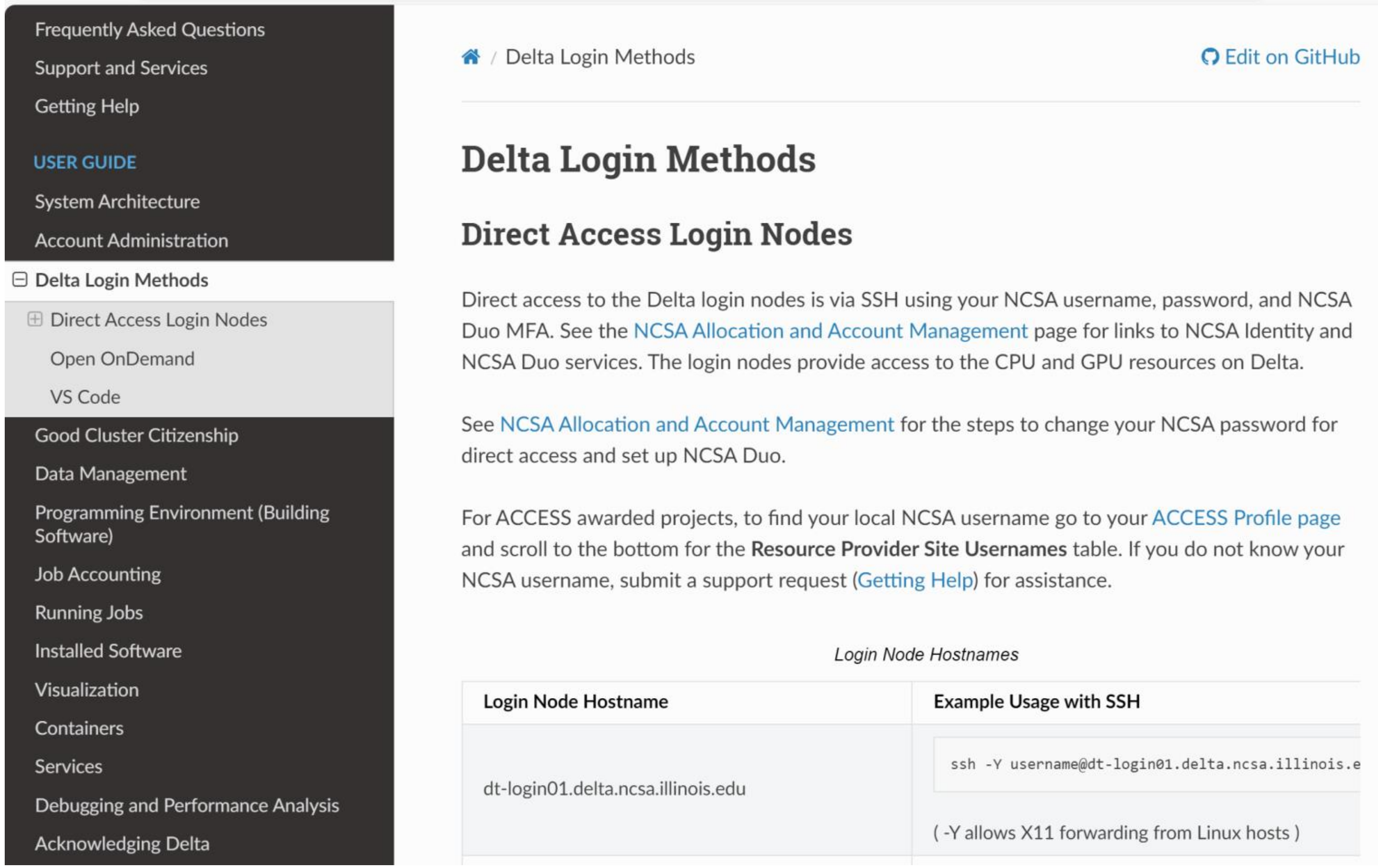
- Home page: https://www.ncsa.illinois.edu/research/project-highlights/delta/

- 100 quad A100 GPU node, each with 4 A100
- 100 quad A40 GPU node, each with 4 A40
- 5 8-way A100 GPU, each with 8 A100
- 1 MI100 node, 8 MI100

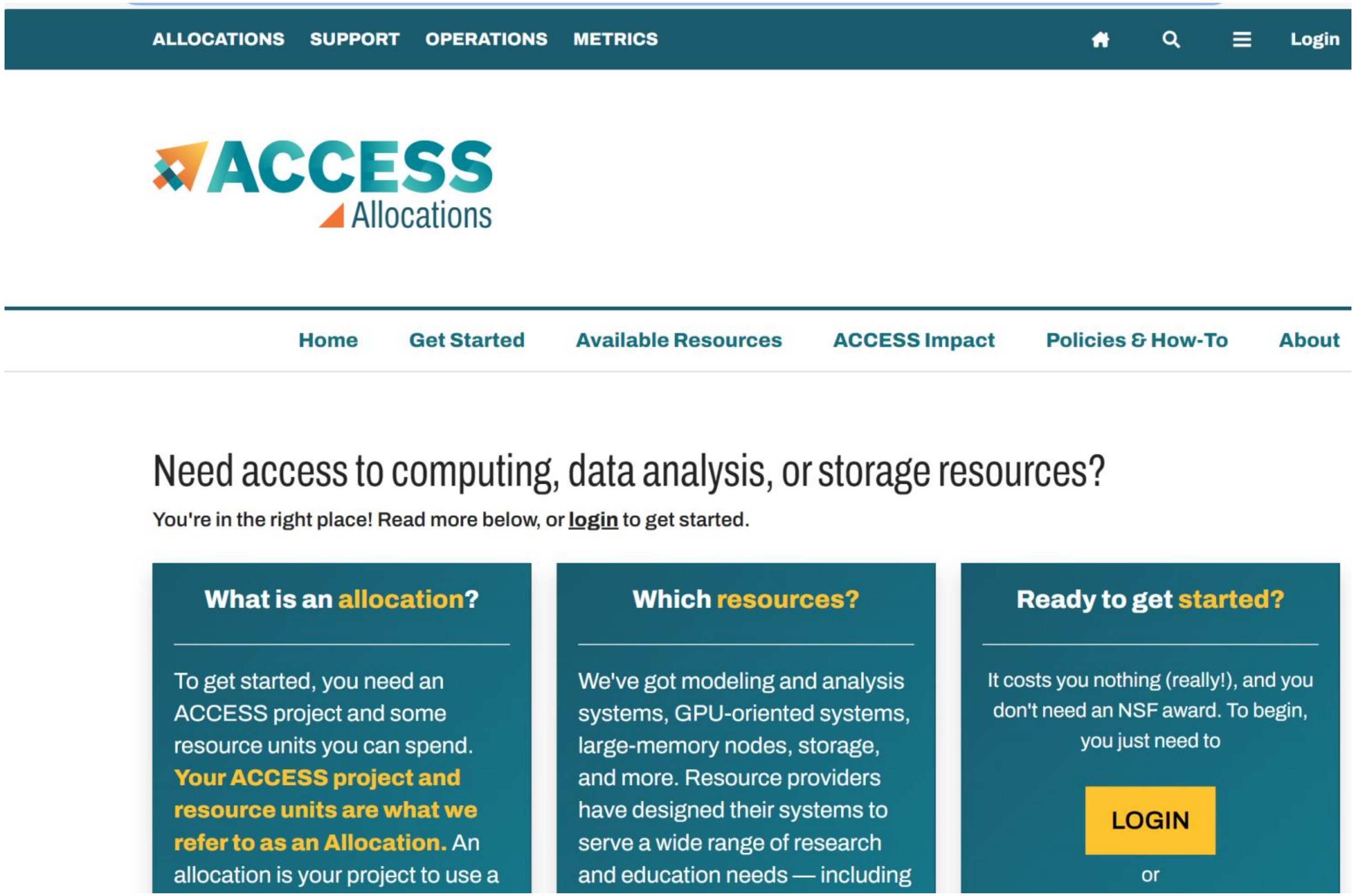


Delta Onboarding Page

- https://docs.ncsa.illinois.edu/systems/delta/en/latest/user_guide/accessing.html

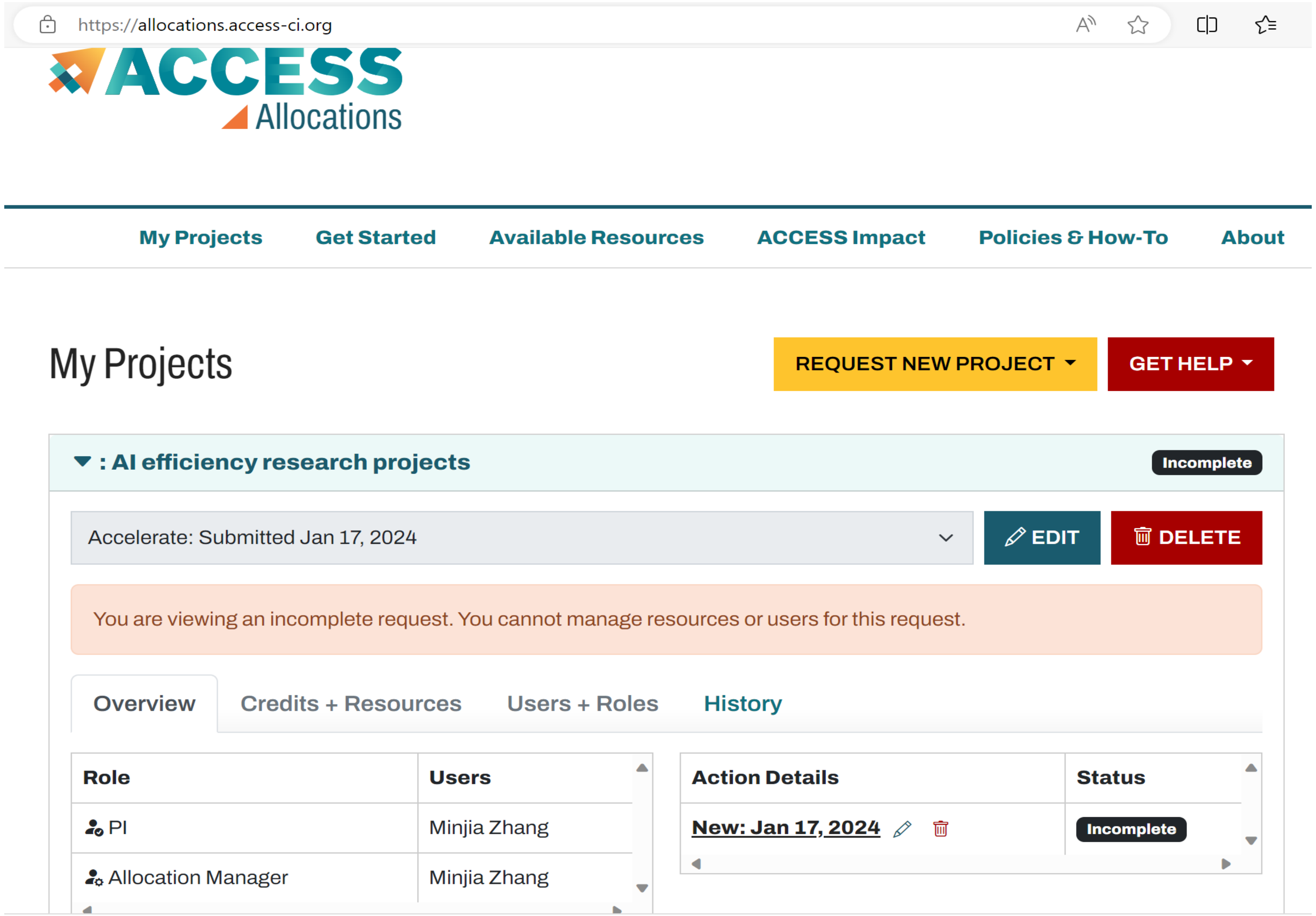


Step 1: Create ACCESS ID



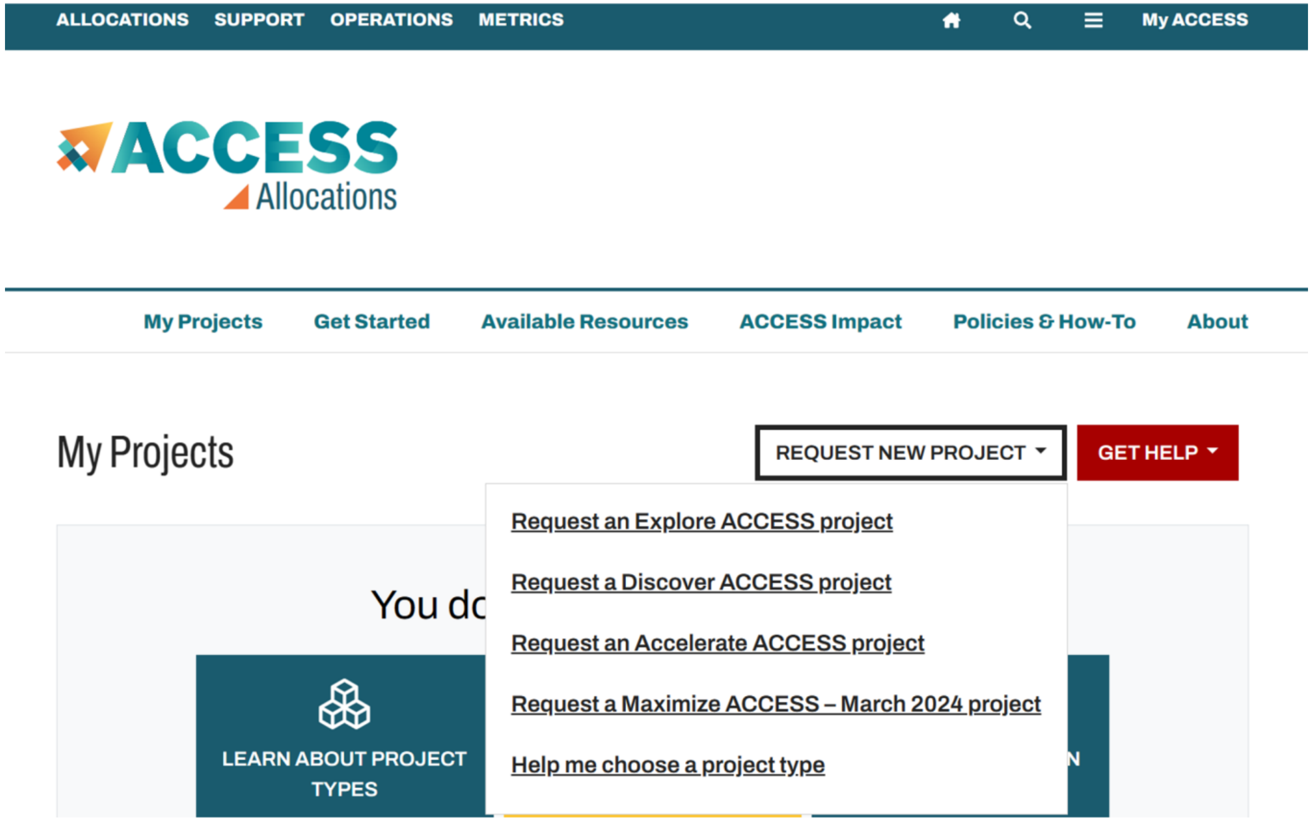
[ACCESS Allocations: ACCESS (access-ci.org)](https://access-ci.org)

Step 2: Submit Resource Requests



Step 2: Submit Resource Requests

- **EXPLORE** — Great for resource evaluation, graduate student projects, small classes and training events, benchmarking, code development and porting, and similar small-scale uses.
- **DISCOVER** — Designed for research grants with modest resource needs, Campus Champions, large classes and training events, NSF graduate fellowships, benchmarking and code testing at scale, and gateway development.
- **ACCELERATE** — Best for experienced users with mid-scale resource needs, consolidating multi-grant programs, collaborative projects, preparing for Maximize ACCESS requests, and gateways with growing communities.
- **MAXIMIZE** — The choice for large-scale research activities that need more resources than the limit for Accelerate ACCESS projects.



Step 3: SSH Login

- Support ssh login
- Maintaining Persistent Sessions: tmux

Delta Login Methods

Direct Access Login Nodes

Direct access to the Delta login nodes is via SSH using your NCSA username, password, and NCSA Duo MFA. See the NCSA Allocation and Account Management page for links to NCSA Identity and NCSA Duo services. The login nodes provide access to the CPU and GPU resources on Delta.

See NCSA Allocation and Account Management for the steps to change your NCSA password for direct access and set up NCSA Duo.

For ACCESS awarded projects, to find your local NCSA username go to your ACCESS Profile page and scroll to the bottom for the **Resource Provider Site Usernames** table. If you do not know your NCSA username, submit a support request (Getting Help) for assistance.

Login Node Hostnames

Login Node Hostname	Example Usage with SSH
dt-login01.delta.ncsa.illinois.edu	`ssh -Y username@dt-login01.delta.ncsa.illinois.e` (-Y allows X11 forwarding from Linux hosts)
dt-login02.delta.ncsa.illinois.edu	`ssh -l username dt-login02.delta.ncsa.illinois.e` (-l username alt. syntax for `user@host`)
login.delta.ncsa.illinois.edu (round robin DNS name for the set of login nodes)	`ssh username@login.delta.ncsa.illinois.edu`

Delta

- Please give it a try
  - Request access (A100, A40, AMD,…)
  - Ssh login
  - Run a small training/inference job, say PyTorch examples
  - Do preliminary performance profiling
- Let me know if you run into any issues
  - Single GPU allocation
  - Multi-GPU allocation
  - Interactive development
  - Isolation
  - Persistent storage