Introduction

This paper is the result of a study item within the IEEE 802 “Network Enhancements for the Next Decade” Industry Connections Activity known as Nendica.

Scope

Study main factors in AI system which impact traffic.
Analyze the major challenges for AI computing network.
Investigate future technologies.
Identify potential standard work.

Purpose

Understand the requirement of AI computing network.
Look for potential standardization opportunity in IEEE802.

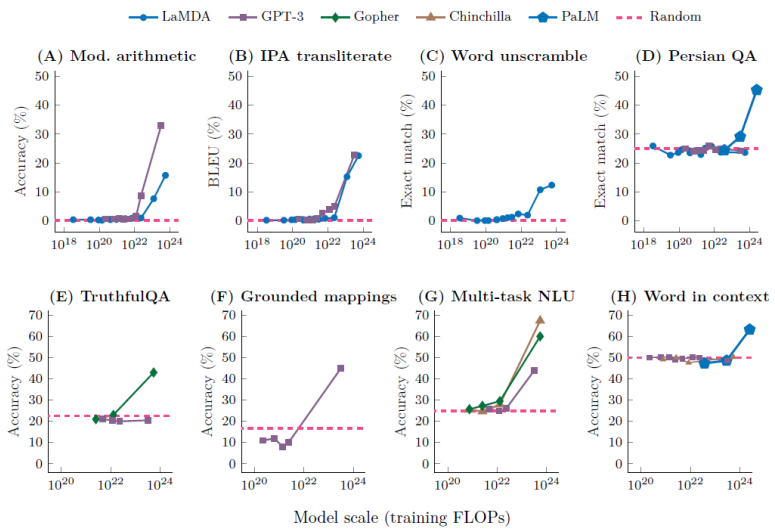
Stepping into the AI era

ChatGPT ignites enthusiasm for AI large models

- ✓ OpenAI announced ChatGPT in November 2022. It gained over 100 million users within 2 months which is the fastest-growing consumer software application in history.
- ✓ What is ChatGPT?
  ChatGPT is short for Chat Generative Pre-trained Transformer. “Chat” is its function. “Generative” represents it uses generative AI technology. ChatGPT has significantly improved in its ability to make conversations, generate high-quality content, and understand language. It is based on GPT-3.5, and further tune the model based on human needs.
- ✓ ChatGPT is a milestone of generative AI. (RNN->Transformer->ChatGPT)
  In the past few years, generative AI has been developing slowly due to the disadvantage of RNN(recurrent neural network). It was not until the "Transformer" architecture emerged in 2017 and solved the problems of the traditional RNN model that generative AI began to develop fast.
  With the geometric growth of model parameters and the exploration of training methods, the emergence of ChatGPT marks that the generic large model breaks through the traditional development dominated by small scale models in the field of NLP.

AI large models show emergent abilities

- ✓ What is emergent ability?
  It is ability that is not present in small scale models but is present in large-scale models. It brings qualitative changes by quantitative changes.
- ✓ With the model scale increase to a certain threshold, performance has a dramatic increase.



From "Emergent Abilities of Large Language Models"

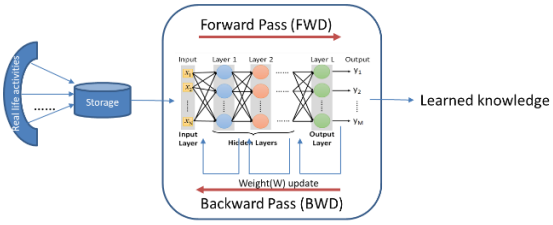
Large AI models require distributed system

<<Introduce AI training background>>

AI working process

✓ working procedure: Input training data->forward pass->backward pass-> gradient update-> forward pass->…

The AI training process involves feeding large amounts of data into a machine learning algorithm. The algorithm adjusts its parameters based on feedback from the data, gradually improving its ability to accurately predict outcomes. This process is repeated multiple times, with the algorithm being tested on new data sets to ensure its accuracy.



Distributed AI system

✓ Why needs distributed system?

A single AI accelerator or a single server with multiple AI accelerator is not capable to train the large models. Take GPT-3(175B) as example, explain the challenges (do not consider model optimization and assume linear performance improvement)

1) Memory: 2.8TB vs. 80GB(A100)
   35 A100 are needed to store the model.
2) Compute power: 430 ZFLOPS vs. 312 TFLOPS (A100 FP16)
   16000 A100 run one day or one A100 runs 43.8 years

So distributed system is imperative to provide enough compute and memory.
Total compute = single accelerator compute * Scale * Efficiency * Availability

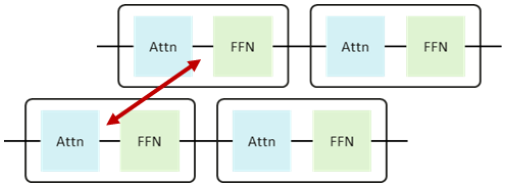
- Scale: number of AI accelerators
- Efficiency: percentage of theoretical peak FLOPS
- Availability: percentage of system working time

✓ Parallel processing in distributed AI system

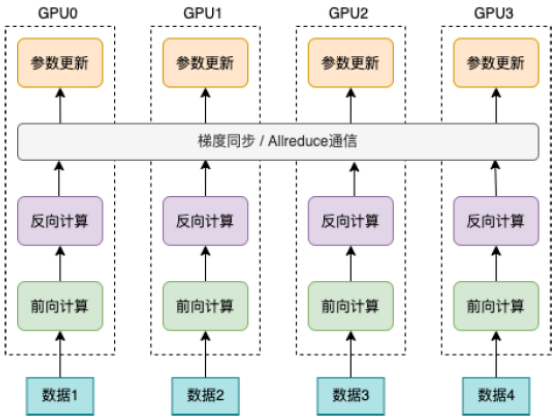
Parallel processing is applied in distributed system for a better performance. Common methods of parallel processing are data parallelism, model parallelism and expert parallelism.

DP:

The training data is divided into multiple batches. Batches are independent from each other. Each batch is processed on a separate accelerator simultaneously. In this approach, each accelerator receives a portion of the data and computes the gradients on its own before they are combined to update the model parameters.



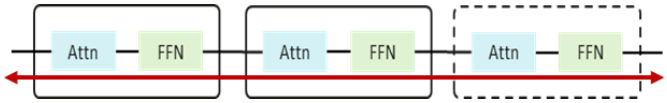
In the forward computation stage, each computing device uses its own data to calculate the loss value. Since the data read by each computing device is different, the loss value obtained on each computing device is often different. In backward computation stage, each computing device calculates the gradient based on the loss value calculated forward, and uses the **AllReduce operation** to calculate the average of the accumulated gradient, thereby ensuring that the gradient value used to update parameters on each computing device is the same. In the parameter update phase, the parameters are updated using the average gradient.



MP: Divide a complete model(neural network) into pieces, and deploy the pieces across multiple accelerators. It solves the problem that large model cannot be stored/trained on a single accelerator. Depending on how the model is cut, it can be further classified as pipeline parallelism and tensor parallelism.

PP: vertical cut of the model

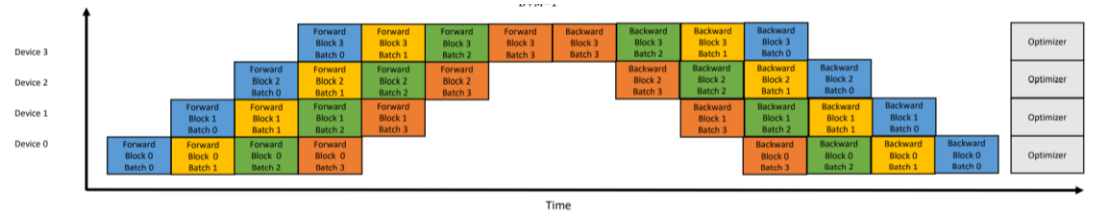
A model is divided into multiple stages and each stage is executed on a separate accelerator. The output of each stage is passed on to the next stage by using **send/recv operation**.



Original PP only solves the memory problem of large model, but cannot accelerate training, because it introduces point to point communication between accelerators, and during the communication, accelerators are idle.

There are many solutions to optimize pipeline parallelism, such as Gpipe, PipeDream, etc. Most of them split minibatch into microbatches, in order to reduce 'bubble' in the pipeline.

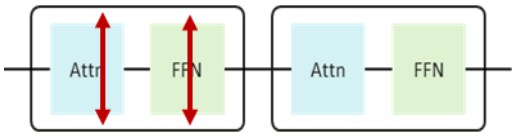
Below is an example of Gpipe. Compared with original PP, it increase accelerator utilization from 1/N to M/(N+M-1), where N is the number of accelerator, M is the number of microbatch.



Huang, Yanping, et al. "Gpipe: Efficient training of giant neural networks using pipeline parallelism." Advances in neural information processing systems 32 (2019).

TP: horizontal cut of the model, intra-layer parallelism

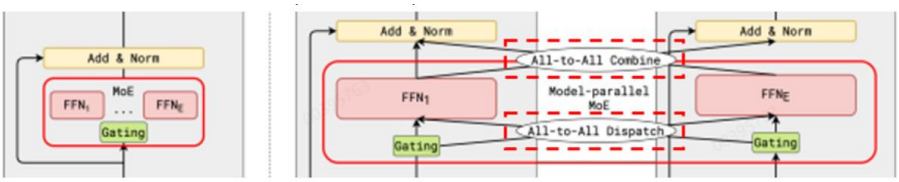
Tensor parallelism focuses on intra-layer parallelization within the model. For example, in Transformer architecture, it uses matrix-matrix multiplication operation. The weight matrix is split along one of its dimensions (usually row or column) and then assigned to multiple accelerators for individual computation. Each accelerator is responsible for only a portion of the entire matrix, and all accelerators synchronize through **AllReduce operation** to merge the result.



MOE

The principle of MOE is to divide complex tasks into multiple sub-tasks and assign them to specific "expert" for processing. It becomes popular in large model due to its scalability and flexibility.

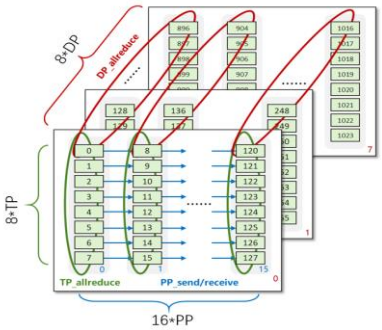
In MOE architecture, it integrates multiple experts and a gate. The gate is responsible for selecting the most suitable expert to deal with specific tasks based on the input data features. Each expert may need to process a portion of all input data, and their outputs need to be aggregated. This introduces **All-to-All** operations.



All-to-All time consumption of MOE layer of DeepSpeed-MoE under 8 A100 GPUs in a single node is 31.18% according to "HetuMoE: An efficient trillion-scale mixture-of-expert distributed training system."

Hybrid parallelism

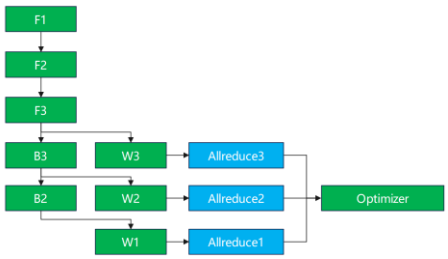
In order to further improve resource utilization and speed up training, those parallel techniques are often combined. For example, in a large-scale AI cluster containing 1024 accelerators, each layer of the model can be split across 8 accelerators through tensor parallelism. The entire model can be divided into 16 pipeline stages according to layer order. The dataset is divided to 8 batches.



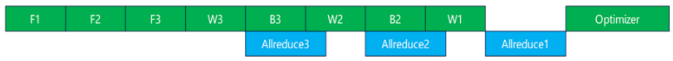
- ✓ **Communications in AI system**
  - ✓ Collective communication
    - What is collective communication

      Communication within a group of nodes (used for average gradients and other subsequent operations).
    - Typical collective communication in AI
      - Allreduce
      - All-to-all
    - Different algorithms to implement collective communication operation
  - ✓ Computing and communication overlapping
    - Communication that is weakly dependent on computing tasks can be overlapped with computing.

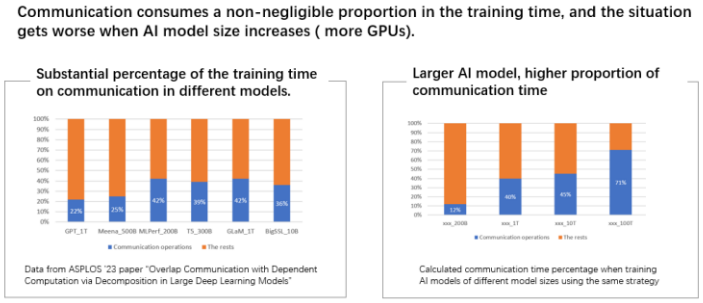
      A Typical case is AllReduce communication in DP. AllReduce is required in backward pass when aggregating the gradients.



Allreduce3 is independent of B3, B2, W2 and W1. Allreduce2 is independent of B2 and W1. So Allreduce3 and Allrduce2 can be overlapped by computing, while Allreduce1 cannot.



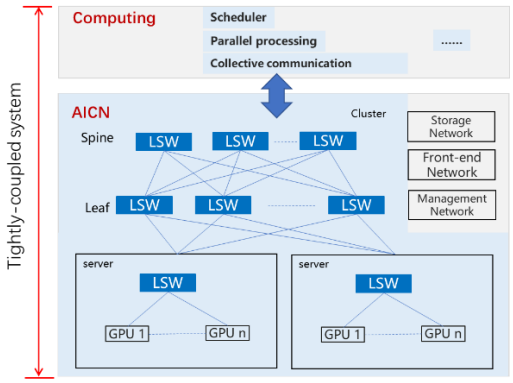
- Communication that is strongly dependent on computing tasks cannot be overlapped with computing.
  Typical cases are 1) Allreduce in tensor parallelism 2) AlltoAll in MOE. In such cases, computing and communication must be executed in sequence. Overhead of communication becomes the main bottleneck.
- Communication time is typically 20%~40 taking into account of overlapping.



The uniqueness of AI computing networks

Systematic view on AICN

✓ AI computing networks interconnect AI accelerators, enabling the distributed system to provide enough compute power and memory for AI workloads. It is tightly coupled with computing.



Traffic injected into AICN is mainly decided by computing scheduler which select communication domain, computing framework which arranges parallelisms, as well as collective communication library which implement collective communication operator.

AICN must efficiently use its resources to complete communication as fast as possible.

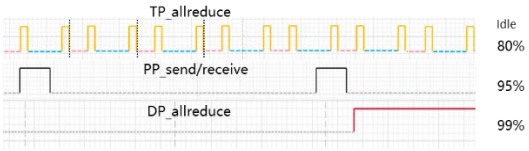
Characteristics of AICN traffic

✓ Characteristics of traffic running on AI computing networks

1) Sparse traffic in space → Regularity of traffic flow
   The communication pair is predictable. The maximum number of connections on an accelerator is TP-1+DP-1+1 if it is TP/DP/PP hybrid deployment.

   Some numbers:
   dense model, 1000s accelerators, 100s B (GPT3): 0.57%~1.5% pairs have traffic
   sparse model, 10000s accelerators, 1000s B(GPT4), 0.024%~0.86% pairs have traffic

2) Sparse traffic in time → square waveform



   TP/DP/PP form different logical planes. Each plane shows periodic bursts of traffic. Link is idle in most of time.

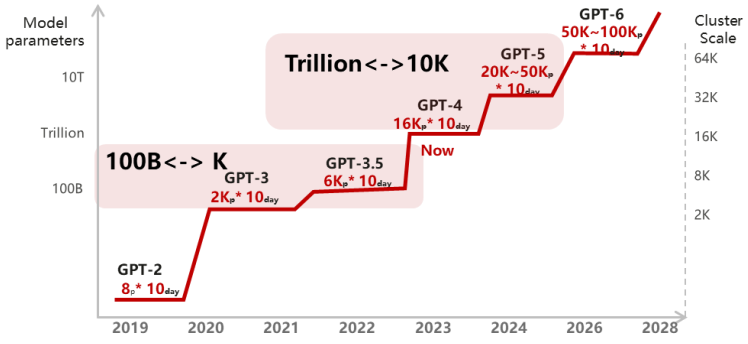
3) Large bandwidth for burst traffic

Parallel Mode	Communication (1 GPU 1 time)
TP	100s GB level
PP	100s MB level
DP	GB level
MOE	GB level

Requirements and Challenges of AI computing Networks

Total compute of distributed AI system = single GPU compute * Scale * Efficiency * Availability

Scale



With 10K->100K connection, there are challenges in terms of network cost and power consumption.

<<show numbers>>

Efficiency

<<large bandwidth is the key>>

✓ Load balancing

How LB benefit improving efficiency

AI workload feature: few Elephant flows with Low entropy.→ECMP works badly→ a more fine-grained load balancing scheme.

Availability

Components in large scale system frequently fail.

✓ Fast Failure recovery

AI fabric's requirement on failure recovery

Future technologies

Standard considerations