Parallelizing GANs

April 18, 2025

1 Introduction

GAN stands for Generative Adversarial Network. It is an approach to generative modeling that creates new data samples resembling the training data. A GAN consists of two neural networks: a generator and a discriminator. During training, the generator and discriminator engage in a competitive game. The generator produces artificial data samples (e.g., fake images) to deceive the discriminator into classifying them as real. In turn, the discriminator attempts to distinguish between real and fake samples. Through repeated iterations, both networks improve their respective roles.

Training GANs is computationally intensive. With large datasets and deep architectures, GAN training can benefit significantly from parallelization on GPUs

2 Methods to parallelize GANs with multiple GPU:

2.1 Data Parallelism

In data parallelism, the dataset is divided into smaller batches. Each GPU processes one batch of data at a time. During this process, each GPU computes the loss and the gradients that indicate how to adjust the model's weights. After all GPUs complete their tasks, the calculated gradients are aggregated and used to update the model weights.

Each GPU maintains an identical copy of the model while working on different data subsets. Communication occurs when gradients computed by each device are synchronized across GPUs to update the model's weights. This typically happens after each batch or group of batches.

Data parallelism is effective when the model is small enough to fit on a single device but the dataset is large. It scales well with the amount of data and improves training efficiency without altering the model structure.

2.2 Model Parallelism

In model parallelism, the model itself is split across multiple GPUs. For instance, different layers of the generator or discriminator may be assigned to different GPUs. As data flows through the model, intermediate outputs (activations) are passed between GPUs.

This approach is useful when the model is too large to fit into the memory of a single GPU. However, it introduces inter-device communication overhead and synchronization delays. A pipeline structure (e.g., Layer 1 on GPU 1, Layer 2 on GPU 2) can cause bottlenecks, as GPUs must wait for each other, reducing overall efficiency.

3 Methods to parallelize GANs with single GPU:

3.1 Batch Parallelism

Batch parallelism accelerates training by processing multiple samples in a dataset simultaneously. It leverages the high parallel compute capabilities of GPUs to apply operations across an entire mini-batch, improving efficiency and throughput.

The batch is split into smaller chunks handled by separate threads within a CUDA kernel. Each thread processes one or more samples, generating data with the generator or computing outputs with the discriminator. Threads work in parallel during both the forward pass (e.g., image generation, score computation) and the backward pass (e.g., gradient calculation). Once all threads complete their computations, the results are combined and used to update the model.

Modern GPUs can launch thousands of threads at once. By feeding a batch to the GPU, threads operate simultaneously on different data portions using CUDA's SIMT (Single Instruction, Multiple Threads) model.

3.2 Vector Parallelism

Vector parallelism exploits the SIMD (Single Instruction, Multiple Data) capabilities of modern GPUs. Instead of processing one data point at a time, operations are vectorized to handle multiple elements in parallel. This is especially valuable in GANs, which frequently involve large matrix operations.

During a forward pass, vectorized operations compute activations across entire layers at once. On the GPU, each thread may process a portion of a batch or a vector segment. This boosts performance by executing multiple calculations concurrently.

Common operations that benefit from vector parallelism include:

- Matrix multiplications
- Convolutions
- Activation functions

These operations are executed via CUDA kernels and optimized libraries such as cuDNN and cuBLAS, which automatically leverage the GPU's parallel hardware to perform tensor-level computations efficiently.

4 Conclusion

Parallelization is essential for efficient GAN training, particularly on large datasets and complex architectures. Data and model parallelism distribute work across multiple GPUs, while batch and vector parallelism optimize computations within a single GPU. Understanding and leveraging these strategies can significantly enhance performance, scalability, and resource utilization during GAN training.