EXAM PROFESSIONAL MACHINE LEARNING ENGINEER TOPIC 1 QUESTION 103 DISCUSSI...

Actual exam question from Google's Professional Machine Learning Engineer
Question #: 103
Topic #: 1
[All Professional Machine Learning Engineer Questions]

You recently developed a deep learning model using Keras, and now you are experimenting with different training strategies. First, you trained the model using a single GPU, but the training process was too slow. Next, you distributed the training across 4 GPUs using tf.distribute.MirroredStrategy (with no other changes), but you did not observe a decrease in training time. What should you do?

A. Distribute the dataset with tf.distribute.Strategy.experimental_distribute_dataset

B. Create a custom training loop.

C. Use a TPU with tf.distribute.TPUStrategy.

D. Increase the batch size.

Show Suggested Answer

by mil_spyro at *Dec. 18, 2022, 8:46 p.m.*

Comments

egdiaa Highly Voted 1 year, 10 months ago

Selected Answer: D

Ans D: Check this link https://www.tensorflow.org/guide/gpu_performance_analysis for details on how to Optimize the performance on the multi-GPU single host

upvoted 10 times

pinimichele01 Most Recent 6 months, 3 weeks ago

Selected Answer: D

when using tf.distribute.MirroredStrategy, TensorFlow automatically takes care of distributing the dataset across the available devices (GPUs in this case).

To make sure that the data is efficiently distributed across the GPUs, you should increase the global batch size. This ensures that each GPU receives a larger batch of data to process, effectively utilizing the additional computational power. The global batch size is the sum of the batch sizes for all devices. For example, if you had a batch size of 64 for a single GPU, you would set the global batch size to 256 (64 * 4) when using 4 GPUs.

upvoted 2 times

pico 11 months, 3 weeks ago

Selected Answer: A

When you distribute the training across multiple GPUs using tf.distribute.MirroredStrategy, the training time may not decrease if the dataset loading and preprocessing become a bottleneck. In this case, option A, distributing the dataset with tf.distribute.Strategy.experimental_distribute_dataset, can help improve the performance.

upvoted 3 times

pico 11 months, 3 weeks ago

option D can be a reasonable step to try, but it's important to carefully monitor the training process, consider memory constraints, and assess the impact on model performance. It might be a good idea to try both option A (distributing the dataset) and option D (increasing the batch size) to see if there is any improvement in training time.

upvoted 1 times

PST21 1 year, 3 months ago

A. Distribute the dataset with tf.distribute.Strategy.experimental_distribute_dataset
When you distribute the training across multiple GPUs using tf.distribute.MirroredStrategy, you need to make sure that the data is also distributed across the GPUs to fully utilize the computational power. By default, the tf.distribute.MirroredStrategy replicates the model and uses synchronous training, but it does not automatically distribute the dataset across the GPUs.

upvoted 1 times

tavva_prudhvi 12 months ago

You are right, However, when using tf.distribute.MirroredStrategy, TensorFlow automatically takes care of distributing the dataset across the available devices (GPUs in this case).

To make sure that the data is efficiently distributed across the GPUs, you should increase the global batch size. This ensures that each GPU receives a larger batch of data to process, effectively utilizing the additional computational power. The global batch size is the sum of the batch sizes for all devices. For example, if you had a batch size of 64 for a single GPU, you would set the global batch size to 256 (64 * 4) when using 4 GPUs.

upvoted 1 times

CloudKida 1 year, 6 months ago

Selected Answer: D

When going from training with a single GPU to multiple GPUs on the same host, ideally you should experience the performance scaling with only the additional overhead of gradient communication and increased host thread utilization. Because of this overhead, you will not have an exact 2x speedup if you move from 1 to 2 GPUs.

Try to maximize the batch size, which will lead to higher device utilization and amortize the costs of communication across multiple GPUs. Using the memory profiler helps get a sense of how close your program is to peak memory utilization. Note that while a higher batch size can affect convergence, this is usually outweighed by the performance benefits.

upvoted 2 times

M25 1 year, 6 months ago

Selected Answer: D

Went with D

upvoted 1 times

tavva_prudhvi 1 year, 7 months ago

Selected Answer: D

If distributing the training across multiple GPUs did not result in a decrease in training time, the issue may be related to the batch size being too small. When using multiple GPUs, each GPU gets a smaller portion of the batch size, which can lead to slower training times due to increased communication overhead. Therefore, increasing the batch size can help utilize the GPUs more efficiently and speed up training.

upvoted 3 times

TNT87 1 year, 8 months ago

Selected Answer: D

Answer D

upvoted 1 times

John_Pongthorn 1 year, 9 months ago

D: it is best https://www.tensorflow.org/guide/distributed_training#use_tfdistributestrategy_with_keras_modelfit
Each epoch will then train faster as you add more GPUs. Typically, you would want to increase your batch size as you add more accelerators,
C is rule out because of GPU
A and B , as reading on
https://www.tensorflow.org/guide/distributed_training#use_tfdistributestrategy_with_custom_training_loops To use custom loop , we have call If you are writing a custom training loop, you will need to call a few more methods, see the guide:

Start by creating a tf.data.Dataset normally.
Use tf.distribute.Strategy.experimental_distribute_dataset to convert a tf.data.Dataset to something that produces "per-replica" values. If you want to
https://www.tensorflow.org/api_docs/python/tf/distribute/Strategy

upvoted 4 times

zeic 1 year, 10 months ago

Selected Answer: D

To speed up the training of the deep learning model, increasing the batch size. When using multiple GPUs with tf.distribute.MirroredStrategy, increasing the batch size can help to better utilize the additional GPUs and potentially reduce the training time. This is because larger batch sizes allow each GPU to process more data in parallel, which can help to improve the efficiency of the training process.

upvoted 1 times

ares81 1 year, 10 months ago

Selected Answer: C

TPUs are Google's specialized ASICs designed to dramatically accelerate machine learning workloads. Hence it should be C.

upvoted 1 times

Nayak8 1 year, 10 months ago

Selected Answer: D

I think it's D

upvoted 1 times

MithunDesai 1 year, 10 months ago

Selected Answer: A

I think its A

upvoted 4 times

hiromi 1 year, 10 months ago

Selected Answer: B

B (not sure)
- https://www.tensorflow.org/guide/keras/writing_a_training_loop_from_scratch
-https://www.tensorflow.org/guide/distributed_training#use_tfdistributestrategy_with_custom_training_loops

upvoted 1 times

hiromi 1 year, 10 months ago

Sorry, ans D (by ediaa link)

upvoted 1 times

hiromi 1 year, 10 months ago

It's should A

upvoted 1 times

mil_spyro 1 year, 10 months ago

Selected Answer: A

I think it's A,

https://www.tensorflow.org/api_docs/python/tf/distribute/Strategy#in_short

upvoted 3 times

Start Learning for free

Social Media

Facebook , Twitter
YouTube , Reddit
Pinterest

We are the biggest and most updated IT certification exam material website.
Using our own resources, we strive to strengthen the IT professionals community for free.