One of the benefits of Cloud TPU is that you can scale training between different sized TPU types. All TPU types use the same data-parallel architecture. The only change is that the parallelism increases from 8 cores to 2048 cores.

To take full advantage of larger numbers of TPUs, you must tune several training task parameters. This document explains some common issues and the changes you need to make in your models.

To achieve linear scaling on larger TPU types, keep the per core batch size the same.

For example if you use a batch size of 1024 on a v2-8, use a batch size of 4096 (4 * 1024) on a v2-32. This fully utilizes the TPU hardware. You can use smaller batch sizes, but your training will not scale linearly if you do so.

Many models include a `train_steps` flag where one step corresponds to a single batch of data. When you increase the batch size, scale down the number of training steps so that the total number of training examples remains the same.

For example if the original configuration was a batch size of 1024 for 1000 steps, you could increase the batch size to 4096 and reduce the steps to 250 (1000 / 4). If your model uses an `epochs` flag, you do not need to scale the number of steps.

Larger batch sizes can change convergence behavior of the model, so you might also tune some hyperparameters.

If you scale up to use larger batch sizes, you might also need to use an optimizer that supports large batch sizes. For example, the reference Resnet (https://github.com/tensorflow/tpu/tree/master/models/official/resnet) model uses the LARSOptimizer (https://www.tensorflow.org/api_docs/python/tf/contrib/opt/LARSOptimizer) for larger TPU types.

In general, the best practice for TPU training is to always use resources in the same region. Resource region is particularly important when using TPU Pods because the rate of data transfer from Google Cloud Storage tends to be higher. Ensure you are using a regional Google Cloud Storage bucket in the same region as the TPU for training datasets and checkpoints.

For single device training, you can specify either the TPU name or an IP address, for example: grpc://1.2.3.4:8470.

For TPU Pods you must use the TPU name so that TensorFlow can discover the IP addresses of all the hosts available for training distribution.

Specifying the IP address for a TPU Pod generates the following error:

If you need to run your model and TPU in different zones or regions, you must provide TPUClusterResolver with the TPU zone.

You can pass the `tpu` and `tpu_zone` flags to reference models or directly set the values for TPUClusterResolver. For example:

A related issue that can occur when using TPUEstimator is not passing in the cluster resolver to `tf.contrib.tpu.RunConfig`. This could happen if RunConfig is initialized with the `master` parameter. Instead, initialize it with the `cluster` parameter.

If you are using a TPU Pod and `tf.Session` without using TPUEstimator, you must configure the session config, as follows:

Evaluation is neither supported nor cost-effective on TPU Pods. Google recommends running evaluation on a single-device TPU using the `mode` flag, which is supported by most models.

For training on a TPU Pod or single-device TPU, set the flag to `train`. For evaluation, set `eval` for single-device TPUs only. You can also use the mode `train_and_eval` on single-device TPUS, but not on TPU Pods.