This document describes the architecture for all of the hardware and software components of the Cloud TPU system.

Tensor Processing Units (TPUs) are Google's custom-developed application-specific integrated circuits (ASICs) used to accelerate machine learning workloads. These TPUs are designed from the ground up with the benefit of Google's deep experience and leadership in machine learning.

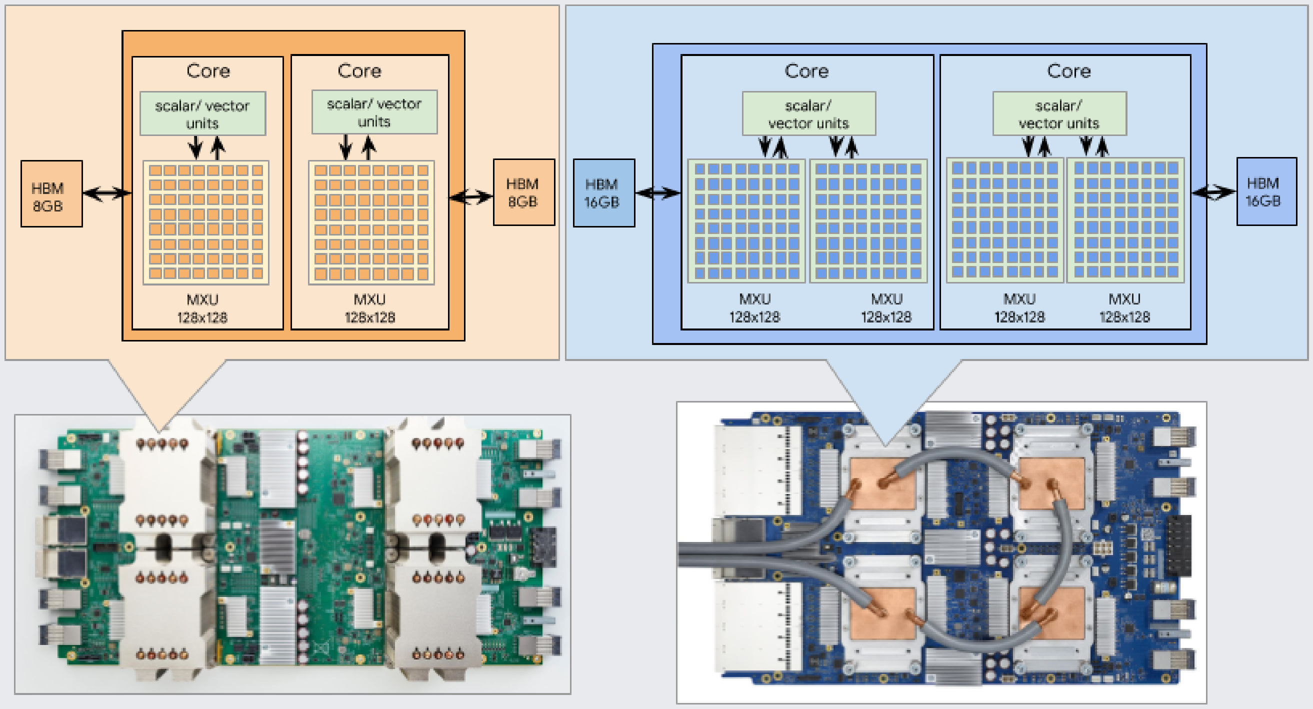
You can use Cloud TPU and TensorFlow (https://www.tensorflow.org/) to run your own machine learning workloads on Google's TPU accelerator hardware (/tpu/). Cloud TPU is designed for maximum performance and flexibility to help researchers, developers, and businesses build TensorFlow compute clusters that can use CPUs, GPUs, and TPUs. High-level Tensorflow APIs make it easy to run replicated models on Cloud TPU hardware.

Your TensorFlow applications can access TPU nodes from containers, instances, or services on Google Cloud. The application requires a connection to your TPU node through your VPC network (/vpc/docs/using-vpc).

Each TPU version defines the specific hardware characteristics of a TPU device. The TPU version defines the architecture for each TPU core, the amount of high-bandwidth memory (HBM) for each TPU core, the interconnects between the cores on each TPU device, and the networking interfaces available for inter-device communication. For example, each TPU version has the following characteristics:

- TPU v2:
  - 8 GiB of HBM for each TPU core
  - One MXU for each TPU core
  - Up to 512 total TPU cores and 4 TiB of total memory in a TPU Pod (#pod)

- TPU v3:
  - 16 GiB of HBM for each TPU core
  - Two MXUs for each TPU core
  - Up to 2048 total TPU cores and 32 TiB of total memory in a TPU Pod (#pod)



TPU v2 - 4 chips, 2 cores per chip

TPU v3 - 4 chips, 2 cores per chip

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 (https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/framework/bfloat16.h) precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision (https://en.wikipedia.org/wiki/Half-precision_floating-point_format) representation.

Each of the cores on a TPU device can execute user computations (XLA ops) independently. High-bandwidth interconnects allow the chips to communicate directly with each other on the TPU device. In a TPU Pod configuration (#pod), dedicated high-speed network interfaces connect multiple TPU devices together to provide a larger number of TPU cores and a larger pool of TPU memory for your machine learning workloads.

The increased FLOPS per core and memory capacity in TPU v3 configurations can improve the performance of your models in the following ways:

- TPU v3 configurations provide significant performance benefits per core for compute-bound models. Memory-bound models on TPU v2 configurations might not achieve this same performance improvement if they are also memory-bound on TPU v3 configurations.
- In cases where data does not fit into memory on TPU v2 configurations, TPU v3 can provide improved performance and reduced recomputation of intermediate values (re-materialization).
- TPU v3 configurations can run new models with batch sizes that did not fit on TPU v2 configurations. For example, TPU v3 might allow deeper ResNets and larger images with RetinaNet.

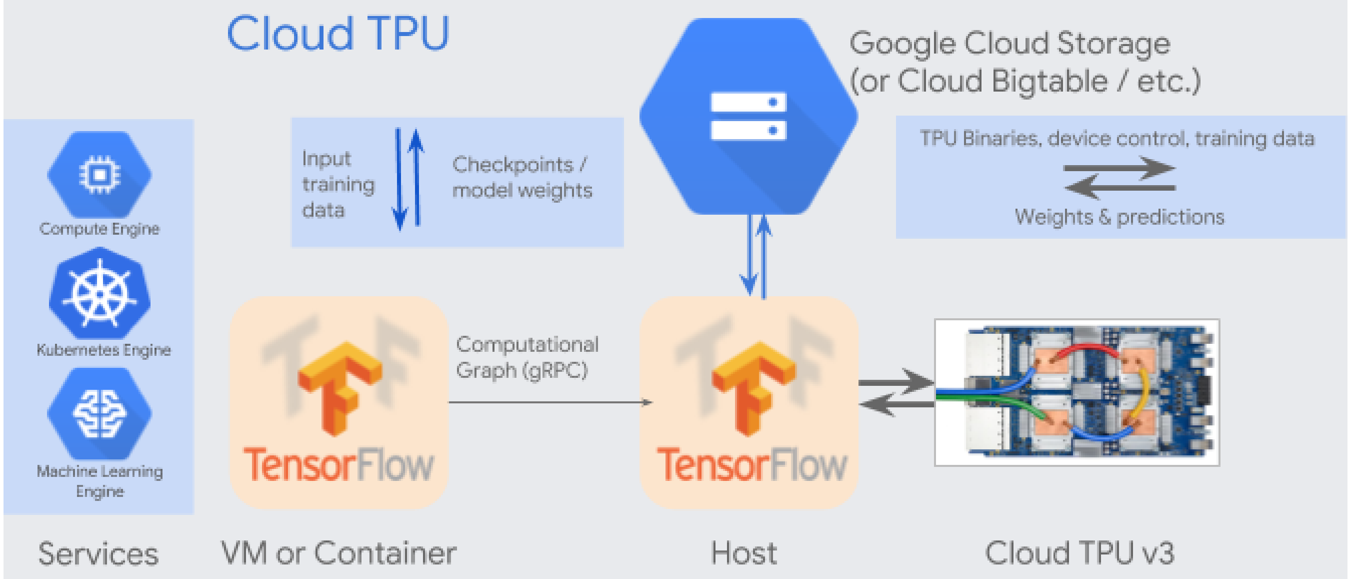
Models that are nearly input-bound ("infeed") on TPU v2 because training steps are waiting for input might also be input-bound with Cloud TPU v3. The pipeline performance guide (https://www.tensorflow.org/performance/datasets_performance) can help you resolve infeed issues.

You can determine if the performance of your model will improve from TPU v3 by running benchmarks on different TPU versions and monitoring the performance using TensorBoard tools (https://cloud.google.com/tpu/docs/tensorboard-setup).

In a Google data center, TPU devices are available in the following configurations for both TPU v2 and TPU v3:

- Single device TPUs (#device), which are individual TPU devices that are not connected to each other over a dedicated high-speed network. You cannot combine multiple single device TPU types to collaborate on a single workload.
- TPU Pods (#pod), which are clusters of TPU devices that are connected to each other over dedicated high-speed networks.

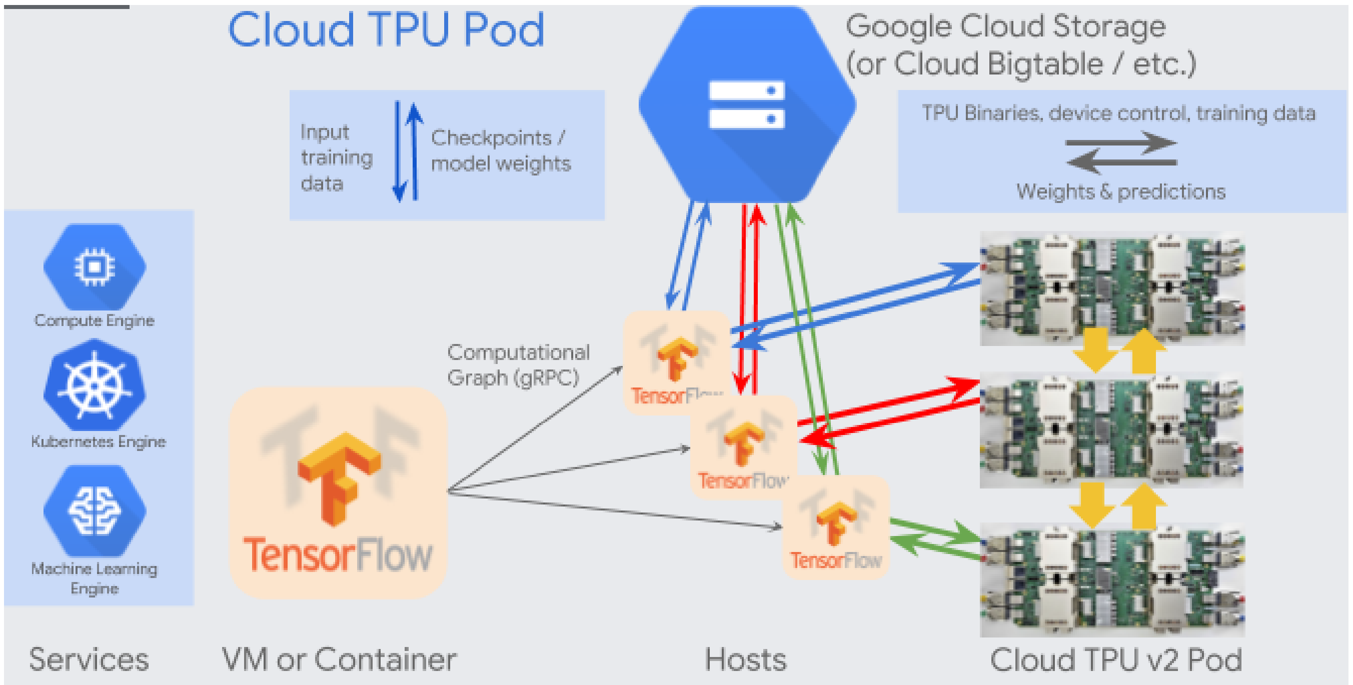
A single-device TPU configuration in a Google data center is one TPU device with no dedicated high-speed network connections to other TPU devices. Your TPU node connects only to this single device.



For single-device TPUs the chips are interconnected on the device so that communication between chips does not require host CPU or host networking resources.

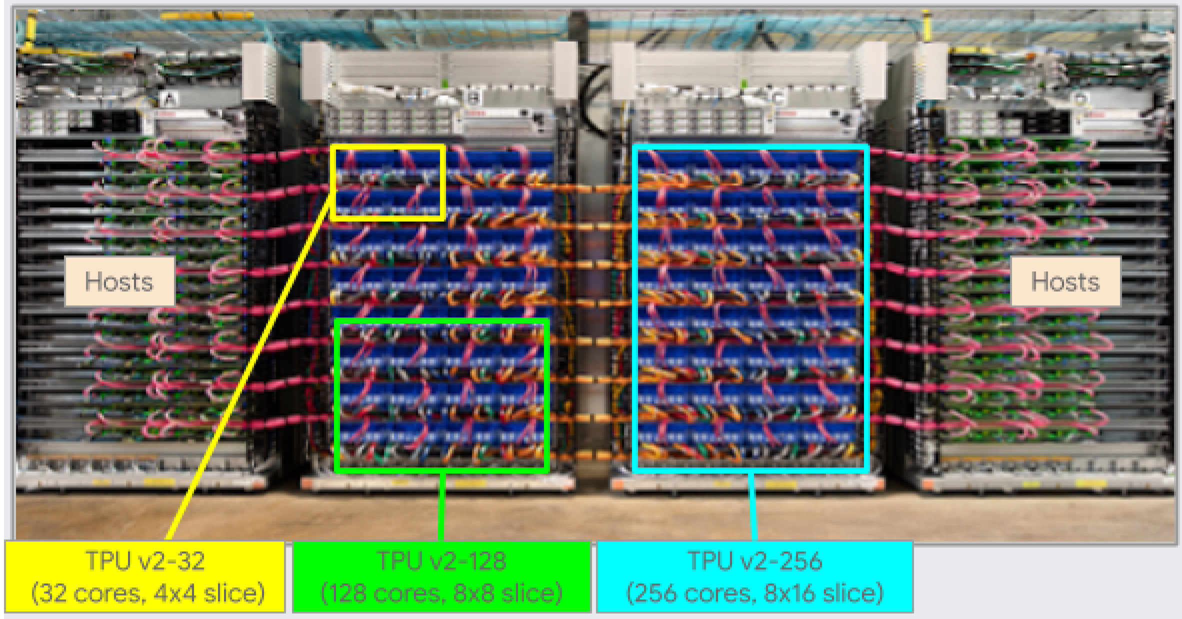
When you create a TPU node, you specify the TPU type. For example, you might specify `v2-8` or `v3-8` to configure your TPU node with a single device. Single-device TPUs are not part of TPU Pod (#pod) configurations and do not occupy a portion of a TPU Pod. Read the TPU types (/tpu/docs/types-zones#device-types) page to see what single-device TPU configurations are available for your TPU nodes.

A TPU pod configuration in a Google data center has multiple TPU devices connected to each other over a dedicated high-speed network connection. The hosts in your TPU node distribute your machine learning workloads across all of the TPU devices.

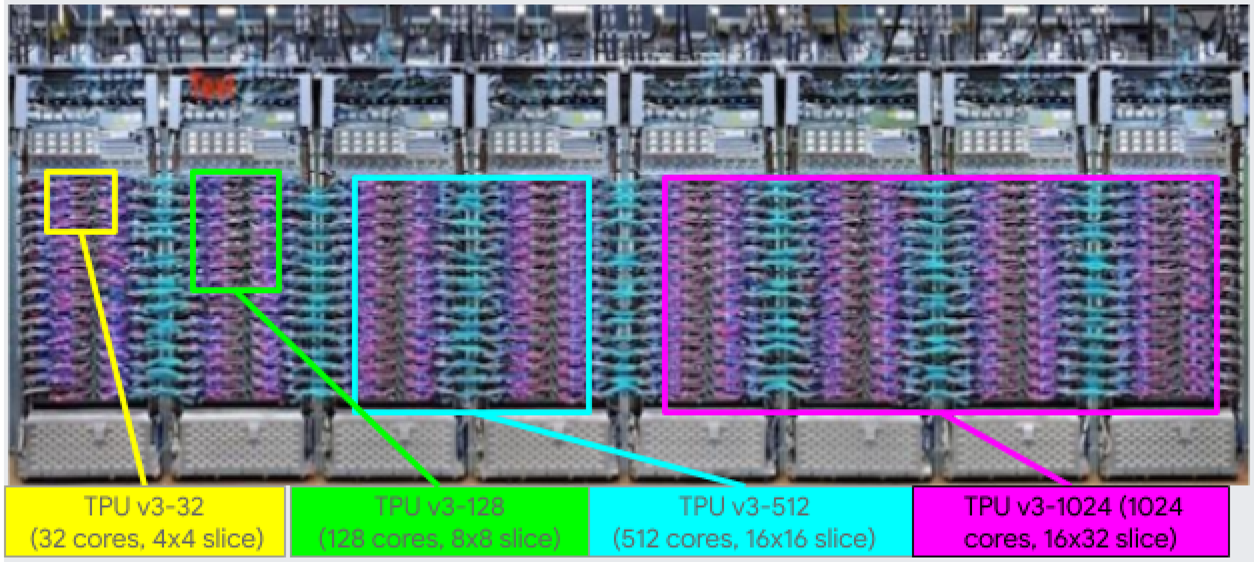


In a TPU Pod, the TPU chips are interconnected on the device so that communication between chips does not require host CPU or host networking resources. Additionally, each of the TPU devices in a TPU Pod are connected to each other over dedicated high-speed networks that also do not require host CPU or host networking resources.

When you create a TPU node, you specify that you want a TPU type that occupies either the full TPU pod or a smaller fraction of that TPU pod. For example, a `v2-512` TPU type occupies a full v2 TPU Pod and a `v2-128` TPU type occupies only 1/4th of a v2 TPU Pod. Read the TPU types (/tpu/docs/types-zones#pod-types) page to see what TPU Pod configurations are available for your TPU nodes.



The v2 TPU Pod provides a maximum configuration of 64 devices for a total 512 TPU v2 cores and 4 TiB of TPU memory.



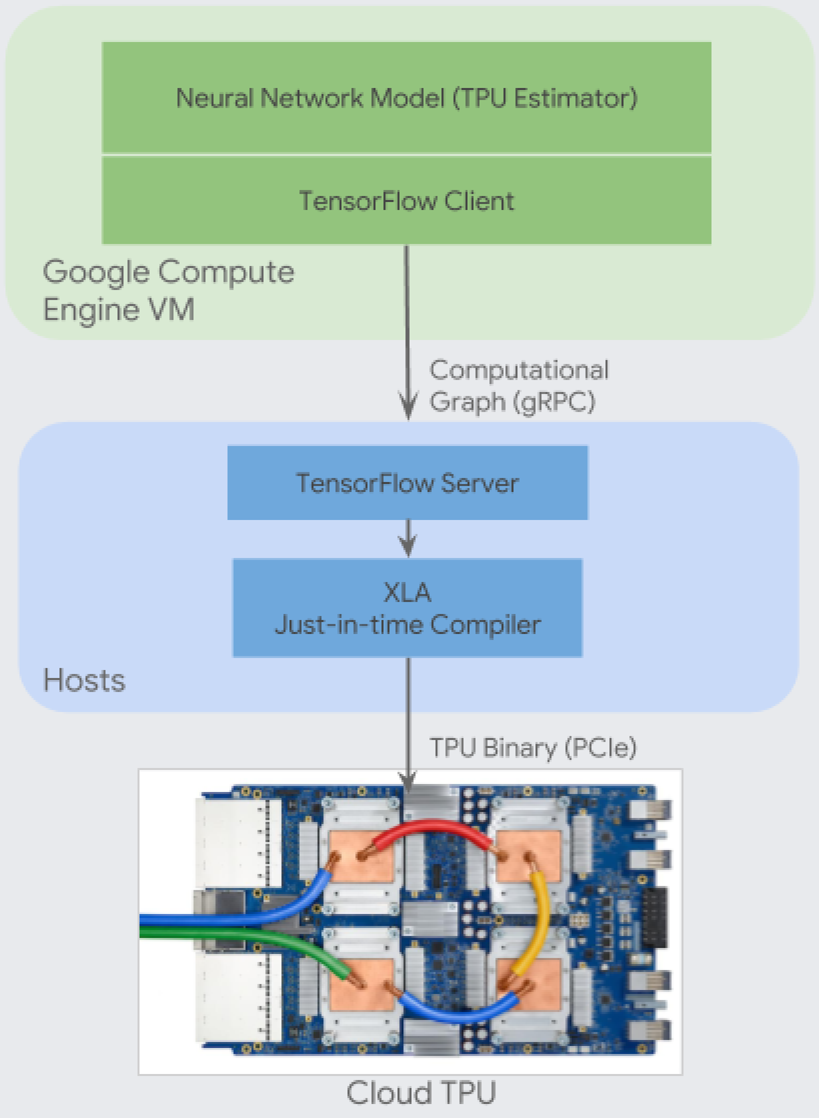
The v3 TPU Pod provides a maximum configuration of 256 devices for a total 2048 TPU v3 cores and 32 TiB of TPU memory.

See the TPU versions (#versions) section to learn more about the architectural differences between different TPU versions.

You can use the Cloud TPU API (/tpu/docs/reference/rest/) to automate TPU management for your TPU nodes regardless of their size. As a result, it is easy to scale up to massive compute clusters, run your workloads, and scale those clusters back down when your workloads are complete. The hardware support built into the chips results in effectively linear performance scaling across a broad range of deep learning workloads. In practice, the Cloud TPU software stack removes the complexity of generating, running, and feeding TPU Cloud programs.

When you run your application, TensorFlow generates a computation graph and sends it to a TPU node over gRPC. The TPU type that you select for your TPU node determines how many devices are available for your workload. The TPU node compiles the computation graph just in time and sends the program binary to one or more TPU devices for execution. Inputs to the model are often stored in Cloud Storage. The TPU node streams the inputs to one or more TPU devices for consumption.

The block diagram below shows the Cloud TPU software architecture, consisting of the neural network model, TPU Estimator and TensorFlow client, TensorFlow server and XLA compiler.



TPU Estimators are a set of high-level APIs that build upon Estimators (https://www.tensorflow.org/programmers_guide/estimators) which simplify building models for Cloud TPU and which extract maximum TPU performance. When writing a neural network model that uses Cloud TPU, you should use the TPU Estimator APIs.

TPU Estimators translate your programs into TensorFlow operations, which are then converted into a computational graph by a TensorFlow client. A TensorFlow client communicates the computational graph to a TensorFlow server.

A TensorFlow server runs on a Cloud TPU server. When the server receives a computational graph from the TensorFlow client, the server performs the following actions:

1. Load inputs from Cloud Storage
2. Partition the graph into portions that can run on a Cloud TPU and those that must run on a CPU
3. Generate XLA operations corresponding to the sub-graph that is to run on Cloud TPU
4. Invoke the XLA compiler

XLA is a just-in-time compiler that takes as input High Level Optimizer (HLO) operations that are produced by the TensorFlow server. XLA generates binary code to be run on Cloud TPU, including orchestration of data from on-chip memory to hardware execution units and inter-chip communication. The generated binary is loaded onto Cloud TPU using PCIe connectivity between the Cloud TPU server and the Cloud TPU and is then launched for execution.

- Read Cloud Tensor Processing Units (TPUs) (/tpu/docs/tpus) to compare Cloud TPU to other processors.
- Read the following documents for help with deciding on a service and on Cloud TPU options:
    - Choosing a Cloud TPU service (/tpu/docs/deciding-tpu-service)
    - Choosing a Cloud TPU configuration (/tpu/docs/types-zones)