

TPU architecture

Tensor Processing Units (TPUs) are application specific integrated circuits (ASICs) designed by Google to accelerate machine learning workloads. Cloud TPU is a Google Cloud service that makes TPUs available as a scalable resource.

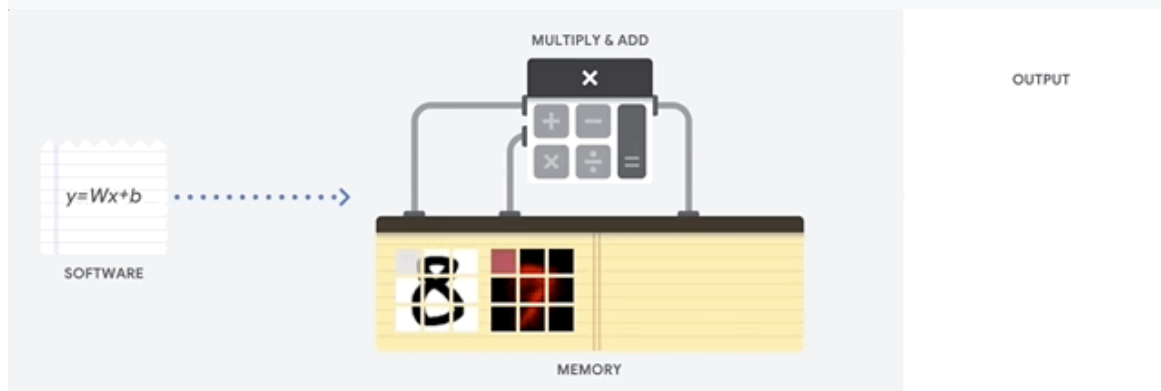
TPUs are designed to perform matrix operations quickly making them ideal for machine learning workloads. You can run machine learning workloads on TPUs using frameworks such as [PyTorch](#) (/tpu/docs/tutorials/pytorch-pod) and [JAX](#) (<https://jax.readthedocs.io/en/latest/>).

How do TPUs work?

To understand how TPUs work, it helps to understand how other accelerators address the computational challenges of training ML models.

How a CPU works

A CPU is a general-purpose processor based on the [von Neumann architecture](#) (https://en.wikipedia.org/wiki/Von_Neumann_architecture). That means a CPU works with software and memory like this:



Note: This animation is designed for conceptual presentation purpose only, and does not reflect the actual behavior of real processors.

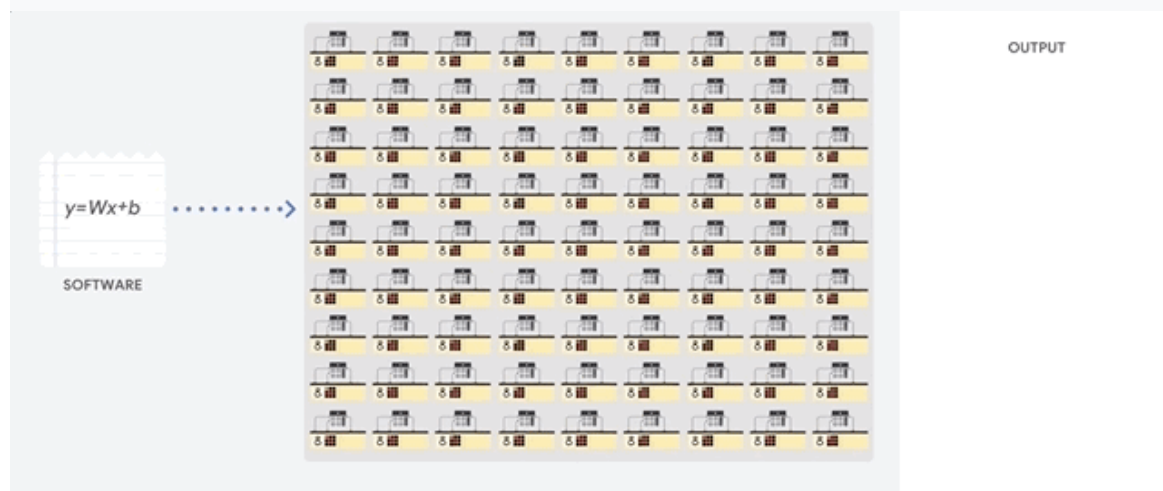
The greatest benefit of CPUs is their flexibility. You can load any kind of software on a CPU for many different types of applications. For example, you can use a CPU for word processing on a PC, controlling rocket engines, executing bank transactions, or classifying images with a neural network.

A CPU loads values from memory, performs a calculation on the values and stores the result back in memory for every calculation. Memory access is slow when compared to the calculation speed and can limit the total throughput of CPUs. This is often referred to as the von Neumann bottleneck

(https://en.wikipedia.org/wiki/Von_Neumann_architecture#von_Neumann_bottleneck).

How a GPU works

To gain higher throughput, GPUs contain thousands of Arithmetic Logic Units (ALUs) in a single processor. A modern GPU usually contains between 2,500–5,000 ALUs. The large number of processors means you can execute thousands of multiplications and additions simultaneously.



Note: This animation is designed for conceptual presentation purpose only, and does not reflect the actual behavior of real processors.

This GPU architecture works well on applications with massive parallelism, such as matrix operations in a neural network. In fact, on a typical training workload for deep learning, a GPU can provide an order of magnitude higher throughput than a CPU.

But, the GPU is still a general-purpose processor that has to support many different applications and software. Therefore, GPUs have the same problem as CPUs. For every calculation in the thousands of ALUs, a GPU must access registers or shared memory to read operands and store the intermediate calculation results.

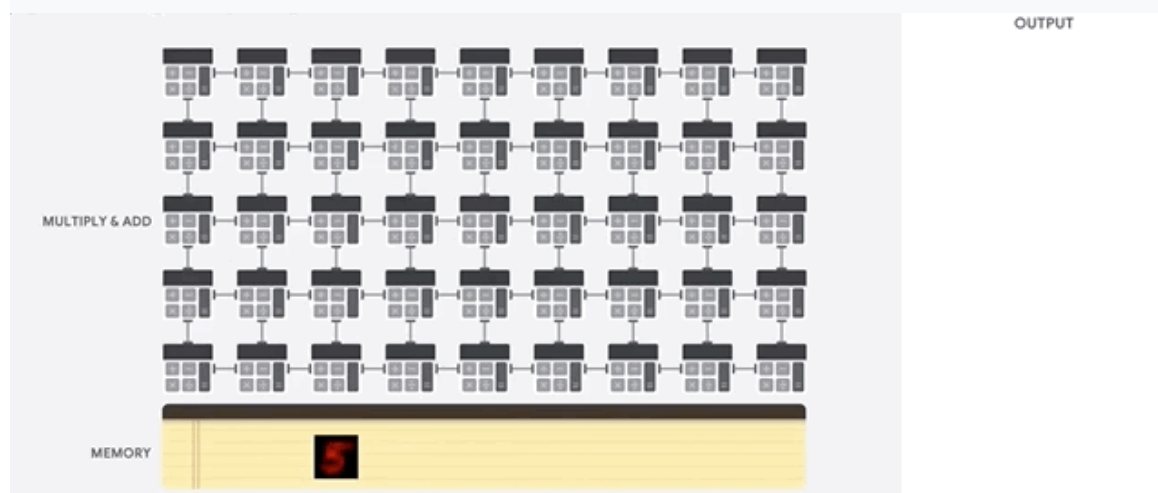
How a TPU works

Google designed Cloud TPUs as a matrix processor specialized for neural network workloads. TPUs can't run word processors, control rocket engines, or execute bank transactions, but they can handle massive matrix operations used in neural networks at fast speeds.

The primary task for TPUs is matrix processing, which is a combination of multiply and accumulate operations. TPUs contain thousands of multiply-accumulators that are directly connected to each other to form a large physical matrix. This is called a systolic array (https://en.wikipedia.org/wiki/Systolic_array) architecture. Cloud TPU v3, contain two systolic arrays of 128 x 128 ALUs, on a single processor.

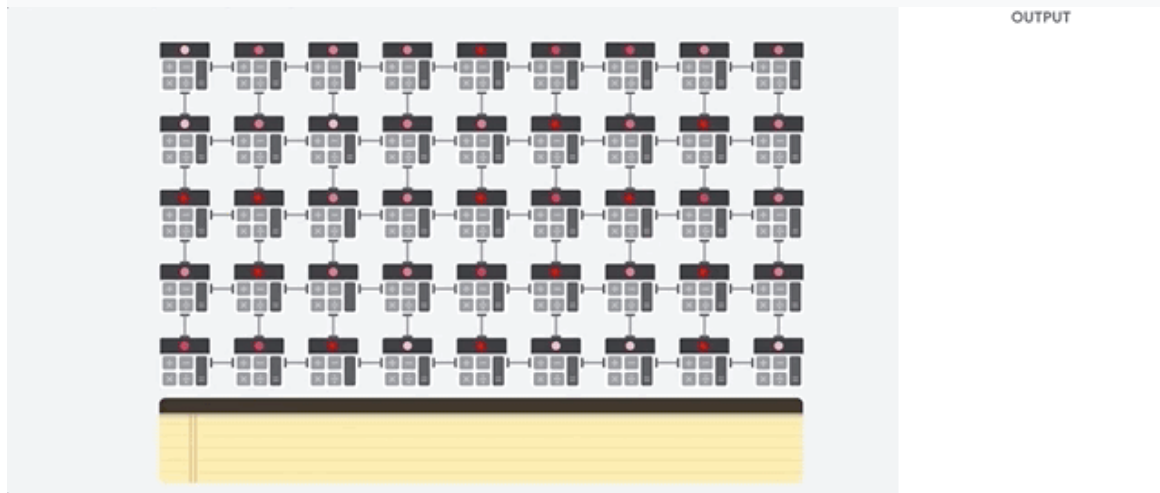
The TPU host streams data into an infeed queue. The TPU loads data from the infeed queue and stores them in High Bandwidth Memory (HBM). When the computation is completed, the TPU loads the results into the outfeed queue. The TPU host then reads the results from the outfeed queue and stores them in the host's memory.

To perform the matrix operations, the TPU loads the parameters from HBM into the Matrix Multiplication Unit (MXU).



Then, the TPU loads data from HBM. As each multiplication is executed, the result is passed to the next multiply-accumulator. The output is the summation of all multiplication results

between the data and parameters. No memory access is required during the matrix multiplication process.



As a result, TPUs can achieve a high-computational throughput on neural network calculations.

TPU system architecture

The following sections describe the key concepts of a TPU system. For more information about common machine learning terms, see [Machine Learning Glossary](https://developers.google.com/machine-learning/glossary) (<https://developers.google.com/machine-learning/glossary>).

If you are new to Cloud TPU, check out the [TPU documentation home page](https://cloud.google.com/tpu/docs) (<https://cloud.google.com/tpu/docs>).

TPU chip

A TPU chip contains one or more TensorCores. The number of TensorCores depends on the version of the TPU chip. Each TensorCore consists of one or more matrix-multiply units (MXUs), a vector unit, and a scalar unit. For more information about TensorCores, see [A Domain-Specific Supercomputer for Training Deep Neural Networks](https://dl.acm.org/doi/pdf/10.1145/3360307) (<https://dl.acm.org/doi/pdf/10.1145/3360307>).

An MXU is composed of either 256 x 256 (TPU v6e and TPU7x) or 128 x 128 (TPU versions prior to v6e) multiply-accumulators in a [systolic array](https://en.wikipedia.org/wiki/Systolic_array) (https://en.wikipedia.org/wiki/Systolic_array). MXUs provide the bulk of the compute power in a TensorCore. Each MXU is capable of

performing 16K multiply-accumulate operations per cycle. All multiplies take bfloat16 (https://en.wikipedia.org/wiki/Bfloat16_floating-point_format) inputs, but all accumulations are performed in FP32 number format.

The vector unit is used for general computation such as activations and softmax. The scalar unit is used for control flow, calculating memory addresses, and other maintenance operations.

TPU Pod

A TPU Pod is a contiguous set of TPUs grouped together over a specialized network. The number of TPU chips in a TPU Pod is dependent on the TPU version.

Slice

A slice is a collection of chips all located inside the same TPU Pod connected by high-speed inter chip interconnects (ICI). Slices are described in terms of chips or TensorCores, depending on the TPU version.

Chip shape and *chip topology* also refer to slice shapes.

Multislice versus single slice

Multislice is a group of slices, extending TPU connectivity beyond the inter-chip interconnect (ICI) connections and leveraging the data-center network (DCN) for transmitting data beyond a slice. Data within each slice is still transmitted by ICI. Using this hybrid connectivity, Multislice enables parallelism across slices and lets you use a greater number of TPU cores for a single job than what a single slice can accommodate.

TPUs can be used to run a job either on a single slice or multiple slices. Refer to the [Multislice introduction](/tpu/docs/multislice-introduction) (/tpu/docs/multislice-introduction) for more details.

TPU cube

A 4x4x4 topology of interconnected TPU chips. This is only applicable to 3D topologies (beginning with TPU v4).

SparseCore

SparseCores are dataflow processors that accelerate models using sparse operations. A primary use case is accelerating recommendation models, which rely heavily on embeddings. v5p and TPU7x have four SparseCores per chip, and v6e has two SparseCores per chip. For an in-depth explanation on how SparseCores can be used, see [A deep dive into SparseCore for Large Embedding Models \(LEM\)](https://openxla.org/xla/sparsecore) (<https://openxla.org/xla/sparsecore>). You control how the XLA compiler uses SparseCores using XLA flags. For more information, see: [TPU XLA flags](https://openxla.org/xla/flags_guidance#tpu_xla_flags) (https://openxla.org/xla/flags_guidance#tpu_xla_flags).

Cloud TPU ICI resiliency

ICI resiliency helps improve fault tolerance of optical links and optical circuit switches (OCS) that connect TPUs between cubes (#cube). (ICI connections within a cube use copper links that are not impacted). ICI resiliency allows ICI connections to be routed around OCS and optical ICI faults. As a result, it improves the scheduling availability of TPU slices, with the trade-off of temporary degradation in ICI performance.

For Cloud TPU v4, v5p, and TPU7x, ICI resiliency is enabled by default for slices that are one cube or larger, for example:

- v5p-128 when specifying accelerator type
- 4x4x4 when specifying accelerator config

TPU versions

The exact architecture of a TPU chip depends on the TPU version that you use. Each TPU version also supports different slice sizes and configurations. For more information about the system architecture and supported configurations, see the following pages:

- [TPU7x \(Ironwood\)](/tpu/docs/tpu7x) (/tpu/docs/tpu7x)
- [TPU v6e](/tpu/docs/v6e) (/tpu/docs/v6e)
- [TPU v5p](/tpu/docs/v5p) (/tpu/docs/v5p)
- [TPU v5e](/tpu/docs/v5e) (/tpu/docs/v5e)
- [TPU v4](/tpu/docs/v4) (/tpu/docs/v4)

- [TPU v3](/tpu/docs/v3) (/tpu/docs/v3)
- [TPU v2](/tpu/docs/v2) (/tpu/docs/v2)

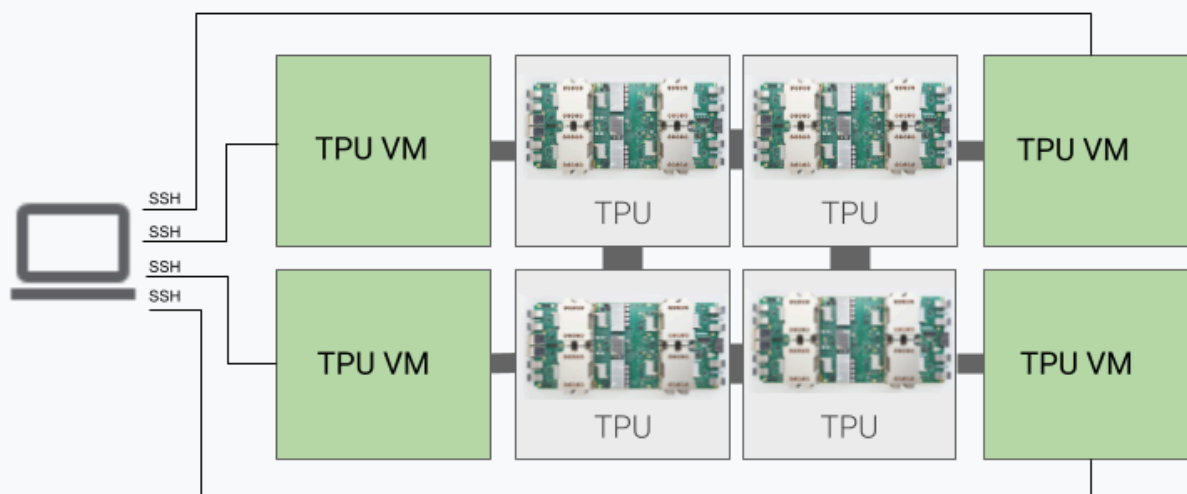
Note: You can run the same code on different versions of TPUs as long as the TPUs have the same number of TensorCores or chips (for example, **v3-128** and **v4-128**). However, if you change to a TPU type with a larger or smaller number of TensorCores or chips, you will need to perform significant tuning and optimization. For more information, see [Training on TPU Pods](/tpu/docs/training-on-tpu-pods) (/tpu/docs/training-on-tpu-pods).

TPU cloud architecture

Google Cloud makes TPUs available as compute resources through TPU VMs. You can directly use TPU VMs for your workloads or use them through Google Kubernetes Engine or Vertex AI. The following sections describe key components of the TPU cloud architecture.

TPU VM architecture

The TPU VM architecture lets you directly connect to the VM physically connected to the TPU device using SSH. A TPU VM, also known as a worker, is a virtual machine running Linux that has access to the underlying TPUs. You have root access to the VM, so you can run arbitrary code. You can access compiler and runtime debug logs and error messages.



Single host, multi host, and sub host

A TPU host is a VM that runs on a physical computer connected to TPU hardware. TPU workloads can use one or more host.

A single-host workload is limited to one TPU VM. A multi-host workload distributes training across multiple TPU VMs. A sub-host workload doesn't use all of the chips on a TPU VM.

TPU topology visualizer

The [TPU topology visualizer](https://tpu-visualizer.uc.r.appspot.com/) (<https://tpu-visualizer.uc.r.appspot.com/>) is a tool that lets you visualize the physical layout of TPUs and their associated networking infrastructure within a physical data center. Use the tool to understand the physical infrastructure layout for different TPU generations and topologies.

What's next

- [Getting started guides](/tpu/docs/quick-starts) (/tpu/docs/quick-starts)
- [Tutorials](/tpu/docs/tutorials) (/tpu/docs/tutorials)

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2026-05-29 UTC.