



PREV

[Chapter 9 Deploying Machine Learning Pipelines](#)[g, Monitoring, and Troubleshooting Mac...](#)

NEXT



Chapter 10 Choosing Training and Serving Infrastructure

Google Cloud Professional Data Engineer Exam objectives covered in this chapter include the following:

- 3. Operationalizing machine learning models
- ✓ 3.3 Choosing the appropriate training and serving infrastructure. Considerations include:
 - Distributed vs. single machine
 - Use of edge compute
 - Hardware accelerators (e.g., GPU, TPU)



Data engineers have a wide array of options for training and serving machine learning models. GCP has a number of serverless options and specialized AI services. We discussed these topics in [Chapter 9](#), “Deploying Machine Learning Pipelines.” This chapter focuses on choosing the appropriate training and serving infrastructure for your needs when serverless or specialized AI services are not a good fit for your requirements.

The Professional Data Engineer exam guide lists three topics relating to choosing training and serving infrastructure with which you should be familiar:

- Use of hardware accelerators
- Distributed and single machine infrastructure
- Use of edge computing for serving machine learning models

We’ll start with considering how to choose between distributed and single machine infrastructure.



Cloud ML Engine has been rebranded by Google as AI Platform. The two terms are used interchangeably here.

Hardware Accelerators

Advances in integrated circuit and instruction set design have led to the development of specialized computing hardware accelerators. These devices offload some of the computing workload from CPUs. Two different types of hardware accelerators are available in GCP:

- Graphics processing units (GPUs)
- Tensor processing units (TPUs)

GRAPHICS PROCESSING UNITS

Graphic processing units (GPUs) are accelerators that have multiple arithmetic logic units (ALUs), which implement adders and multipliers. Modern GPUs have thousands of ALUs. This architecture is well suited to workloads that benefit from massive parallelization, such as training deep learning models. GPUs typically increase training performance on deep learning models by a factor of 10.

GPUs and CPUs share a common design pattern: calculations use registers or shared memory for intermediate calculation results. This can lead to a von Neumann bottleneck, which is the limited data rate between a processor and memory, and slow processing. An alternative design is used by TPUs.

TENSOR PROCESSING UNITS

Tensor processing units (TPUs) are specialized accelerators based on ASICs and created by Google to improve the training of deep neural networks. These accelerators are designed for the TensorFlow framework. The design of TPUs is proprietary and available only through Google's cloud services or on Google's Edge TPU devices.

Compared to GPUs, TPUs perform low-precision computation with as little as 8-bit precision. Using low-precision computation allows for high throughput. In one benchmark, 1 TPU was 27 times faster at training than 8 GPUs.

TPUs are designed to scale horizontally using TPU pods, which have hundreds of TPUs functioning together. TPUs are grouped into pods, which are a two-dimensional mesh network that presents the TPUs as a single programmable resource.

TPUs reduce the impact of the von Neumann bottleneck by implementing matrix multiplication in the processor. This means that TPUs are not used for general-purpose computing tasks but are better suited than GPUs for performing large-scale multiplications and additions, which are needed for training deep learning models.

TPUs can be used with Compute Engine, Kubernetes Engine, and AI Engine.

CHOOSING BETWEEN CPUS, GPUS, AND TPUS

CPUs, GPUs, and TPUs can be used for machine learning, but Google has provided guidelines for choosing the best option for your workload at <https://cloud.google.com/tpu/docs/tpus>.

CPUs are recommended for the following:

- Prototyping
- Simple models that train relatively quickly
- Models that heavily use custom TensorFlow operations written in C++
- Models that are limited by available I/O or network bandwidth of the host server

GPUs are recommended for the following:

- Models that are not built on TensorFlow
- Models with inaccessible source code or code that is difficult to change
- Models that have some custom TensorFlow operations that must run at least partially on CPUs
- Models that use TensorFlow operations not supported on TPUs; available operations are listed at <https://cloud.google.com/tpu/docs/tensorflow-ops>
- Medium-to-large models that can be trained with large batch sizes

TPUs are recommended for the following:

- Models dominated by matrix multiplications
- Models without custom TensorFlow operations inside the main training loop
- Models that can take weeks or months to train on CPUs or GPUs
- Large and very large models that can be trained in very large batch sizes

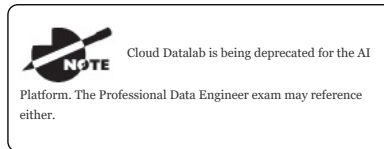
TPUs are not recommended for workloads that require high-precision arithmetic.

Distributed and Single Machine Infrastructure

When discussing machine learning infrastructure, it is helpful to distinguish infrastructure for training machine learning models and infrastructure for serving machine learning models. Training is a compute-intensive operation, and having sufficient computing resources can mean the difference between models that take days to train versus hours to do so. Serving machine learning models—that is, deploying them in production to make predictions—is less CPU intensive. In fact, the widely used deep learning platform TensorFlow includes TensorFlow Lite, which is a framework for running machine learning models on mobile and IoT devices.

SINGLE MACHINE MODEL TRAINING

Single machines are useful for training small models, developing machine learning applications, and exploring data using Jupyter Notebooks or related tools. Cloud Datalab, for example, runs instances in Compute Engine virtual machines (VMs).



The ability to use a single instance is constrained by the available CPUs, GPUs, and memory. With GCE, you have the option of scaling the size of the virtual machine. Currently, the largest machine type, m1-ultramem-160, provides 160 vCPUs and 3.75 TB of memory.

You also have the option of offloading some of the training load from CPUs to GPUs. GPUs have high-bandwidth memory and typically outperform CPUs on floating-point operations. GCP uses NVIDIA GPUs. NVIDIA is the creator of CUDA, a parallel computing platform that facilitates the use of GPUs. As of this writing, GCP offers five GPU types:

- NVIDIA Tesla K80
- NVIDIA Tesla P4
- NVIDIA Tesla T4
- NVIDIA Tesla V100
- NVIDIA Tesla P100

The NVIDIA Tesla K80 can perform over 2.9 teraflop double-precision and over 8.7 teraflops (TFLOPS) single-precision operations and has 24 GB of memory.

The NVIDIA Tesla P4 is optimized for deep learning, and it provides 5.5 TFLOPS on single-precision operations and 22 tera operations per second (TOPS) on integer operations. It has 8 GB of memory.

The NVIDIA Tesla T4 is also well suited to deep learning, and it provides over 8 TFLOPS on single-precision operations and up to 130 TOPS on int8 operations.

The NVIDIA Tesla V100 is a GPU with the equivalent performance of 32 CPUs and the ability to perform 14 TFLOPS.

Compute Engine currently supports attaching up to 8 GPUs to a single instance.

Single machines can be used to train models, as long as the resources, in terms of CPU, GPU, and memory, are sufficient for the volume of data and type of algorithm used. Of course, any time that you are using a single machine, that machine is a potential single point of failure (SPOF).

DISTRIBUTED MODEL TRAINING

Distributing model training over a group of servers provides for scalability and improved availability. There are a variety of ways to use distributed infrastructure, and the best choice for you depends on your specific requirements and development practices.

One way to distribute training is to use machine learning frameworks that are designed to run in a distributed environment.

TensorFlow, the deep learning framework created by Google and released as open source, allows developers to specify a distributed computing strategy when training a model. For example, TensorFlow supports both synchronous and asynchronous training. Synchronous training has all worker training on different subsets of input data and incrementally combines results. In asynchronous training, workers operate independently and update variables asynchronously.

Here are some examples of TensorFlow strategies for distributed training:

MirroredStrategy This strategy supports synchronous distributed training on multiple GPUs on one machine. Each variable is mirrored across all GPUs.

CentralStorageStrategy This strategy is a synchronous training technique in which variables are not mirrored and both CPU and GPUs are used.

MultiWorkerMirroredStrategy This strategy is like the mirrored strategy, but it works across multiple nodes.

TPUStrategy This strategy runs synchronous training on TPUs.

TensorFlow training can run on Kubernetes Engine using the TensorFlow Training Job (TFJob) custom resource. For more on TensorFlow's distributed processing model, see

www.tensorflow.org/guide/distributed_training
(http://www.tensorflow.org/guide/distributed_training).

Alternatively, you can use distributed infrastructure to build multiple models in parallel. For example, you may have a container configured with machine learning tools that is running on GKE. You could use that container to generate models that are trained with different hyperparameters. Each of those models could be trained in parallel, with GKE managing the deployment of pods to meet the workload.

SERVING MODELS

Serving a machine learning model is the process of making the model available to make predictions for other services. When serving models, you need to keep in mind the following:

- Latency
- Scalability
- Version management
- Online versus batch prediction

The *latency* of model predictions will be largely a function of the resources available to execute the model, the type of model, and the amount of preprocessing required. Executing a model and training a model are significantly different. For example, when building a decision tree, an algorithm will consider many instances and features. When the decision tree is used to make predictions, the model makes a series of evaluations using data passed into the model. The number of evaluations will be a function of the depth of the tree, but it will be significantly less work than building the decision tree.

Serving models from a centralized location, such as a data center, can introduce latency because input data and results are sent over the network. If an application needs real-time results, it is better to serve the model closer to where it is needed, such as an edge or IoT device. (See the next section for more on edge computing.)

Models will need to scale to meet workloads. Since this chapter is about infrastructure choices, we will focus on using Compute Engine and Kubernetes Engine. With Compute Engine, you can configure an instance group with a minimum and maximum number of VMs in the group. Compute Engine will automatically scale up or down depending on workload. Kubernetes Engine will likewise scale the number of pods running machine learning models according to workload.

Version control of machine learning models is similar to version management for other software. When you serve a model, you may want to route a small amount of traffic to a new model and the rest of the traffic to the old model. This will give you an opportunity to see how the model performs in production before rolling it out to all users. If the model performs well, you can route all traffic to the new model. Other deployment strategies, such as incremental rollout, can be used as well.

Inference may be performed in batches or online. Batch mode makes predictions for multiple instances at one time. This is useful when you have a large number of instances to evaluate and the evaluation does not have to be done in real time. For example, a retailer that categorizes customers into those likely to leave for a competitor, which is known as “churning,” can use batch mode predictions. This is because the response to the classification process is not needed immediately. Instead, the retailer may send a series of customized offers over several weeks to that customer.

A model that predicts fraudulent credit card transactions, however, would need to execute in online mode. In that case, the model would need to be exposed using an API that applications could call to receive a real-time evaluation.

Edge Computing with GCP

Edge computing is the practice of moving compute and storage resources closer to the location at which it is needed, such as on a factory floor or in a field on a farm.

EDGE COMPUTING OVERVIEW

An early and conceptually simpler forerunner of edge computing is a *content distribution network (CDN)*. CDNs store data at locations close to the users of the data. For example, as shown in [Figure 10.1](#), a global news organization might maintain CDN endpoints across the globe so that users can retrieve content from the closest endpoint rather than from a centralized system. In this use case, edge computing reduces latency for users retrieving content. This is a commonly considered factor when adopting edge computing.



Figure 10.1 CDNs distribute content to servers around the globe.

Edge computing devices can be relatively simple IoT devices, such as sensors with a small amount of memory and limited processing power. This type of device could be useful when the data processing load is light. If all raw data does not need to be stored in a central storage system, then there is no need to transmit all of it from the edge to the central storage system. Instead, the processing that is needed can be done on an edge device, and the results can be sent to the central storage system. [Figure 10.2](#), for example, shows a set of sensors and an edge computing device that can process data locally and then send summary data to the cloud.

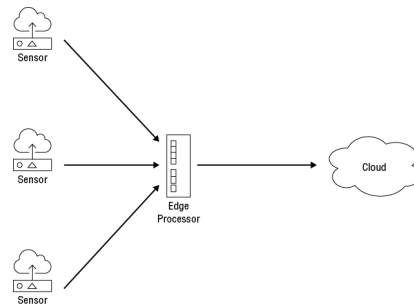


Figure 10.2 Edge computing can be used to process data locally and then send summary data to the cloud.

Edge computing is used when low-latency data processing is needed—for example, to control machinery such as autonomous vehicles or manufacturing equipment. For instance, an edge device could monitor the sounds made by manufacturing equipment to detect when a part does not correctly click into place.

To enable edge computing, the system architecture has to be designed to provide compute, storage, and networking capabilities at the edge while services run in the cloud or an on-premises data center for the centralized management of devices and centrally stored data.

Google Cloud's edge computing offerings include the Edge TPU and the Cloud IoT Edge services.

EDGE COMPUTING COMPONENTS AND PROCESSES

Google Cloud IoT Edge computing consists of three basic components:

- Edge device, which interacts with and senses its environment
- Gateway device, which processes data on behalf of IoT devices
- Cloud platform, such as GCP

Edge devices provide three kinds of data:

- Metadata, such as the device ID, model number, and serial number
- State information about the status of the device
- Telemetry, which is data collected by the device

Before a device is incorporated into an IoT processing system, it must be provisioned and configured. Provisioning includes bootstrapping the device and collecting basic identifying information. For security, the device should be authenticated using a token, key, or other mechanism. The device should also receive authorizations, which define the operations the device is allowed to perform. The device should also be tracked in a central device registry.

After a device is provisioned and it starts collecting data, the data is then processed on the device. The kinds of operations performed can include converting data formats, validating data with business logic rules, summarizing data, and enhancing datasets with additional attributes.

After local processing, the data is transmitted to a gateway. Gateways can manage network traffic across protocols. For example, some IoT devices may not have full network stacks and may communicate with the gateway over other protocols, such as Bluetooth. The gateway in that case receives data over Bluetooth and transmits it to a cloud service over a TCP/IP network.

Data sent to the cloud is ingested by one of a few different kinds of services. In GCP, IoT data can be ingested using the following services:

- Cloud Pub/Sub for scalable ingestion
- IoT Core MQTT, a Message Queue Telemetry Transport broker that uses an industry-standard binary protocol
- Stackdriver Monitoring and Stackdriver Logging, which can also be used to ingest and process metrics and log messages

AI Platform is a managed service that can be used to support deploying machine learning models to the edge. It addresses several factors that you need to consider when making inferences at the edge, including the following:

- Serving prediction
- Restricting access to functionality based on authorizations
- Enforcing quotas on users
- Managing API keys
- Deploying new models

AI Platform also exposes REST APIs so that models can be used for online predictions.

EDGE TPU

Edge TPU is a hardware device available from Google for implementing edge computing. Edge TPU performs inference, so it is used to serve models. Training models occurs in the cloud or other centralized comput-

ing service. This device is an application-specific integrated circuit (ASIC) designed for running AI services at the edge. Edge TPU is designed to work with Cloud TPU and Google Cloud services. In addition to the hardware, Edge TPU includes software and AI algorithms. Here are some use cases for Edge TPU:

- Image processing in a manufacturing environment
- Robotic control
- Anomaly detection in time series data
- Voice recognition and response

The TensorFlow Lite and Android Neural Network API machine learning frameworks can be used to run models on edge devices.

CLOUD IOT

Cloud IoT is Google's managed service for IoT services. The platform provides services for integrating edge computing with centralized processing services. Device data is captured by the Cloud IoT Core service, which can then publish the data to Cloud Pub/Sub for streaming analytics. Data can also be stored in BigQuery for analysis or used for training new machine learning models in Cloud ML. Data provided through Cloud IoT can also be used to trigger Cloud Functions and associated workflows.

Figure 10.3 shows the reference architecture for Cloud IoT.

The reference architecture for Cloud IoT includes the following:

- Cloud IoT Core for device management
- Cloud ML Engine for training and deploying machine learning models
- Cloud Dataflow for streaming and batch analytics
- Cloud Pub/Sub for ingestion
- BigQuery for data warehousing and ad hoc querying

Other GCP services can be used as well, including Cloud Storage for archival storage and Cloud Firestore for storing semi-structured data that will be queried.

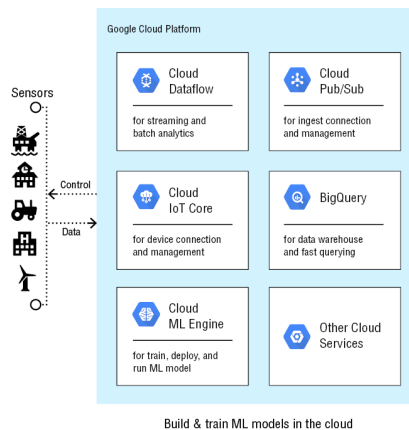


Figure 10.3 Reference architecture for Cloud IoT

Source: <https://cloud.google.com/solutions/iot/>

Exam Essentials

Understand that single machines are useful for training small models. This includes when you are developing machine learning applications or exploring data using Jupyter Notebooks or related tools. Cloud Datalab, for example, runs instances in Compute Engine virtual machines.

Know that you also have the option of offloading some of the training load from CPUs to GPUs. GPUs have high-bandwidth memory and typically outperform CPUs on floating-point operations. GCP uses NVIDIA GPUs, and NVIDIA is the creator of CUDA, a parallel computing platform that facilitates the use of GPUs.

Know that distributing model training over a group of servers provides for scalability and improved availability. There are a variety of ways to use distributed infrastructure, and the best choice for you will depend on your specific requirements and development practices. One way to distribute training is to use machine learning frameworks that are designed to run in a distributed environment, such as TensorFlow.

Understand that serving a machine learning model is the process of making the model available to make predictions for other services. When serving models, you need to consider latency, scalability, and version management. Serving models from a centralized location, such as a data center, can introduce latency because input data and results are sent over the network. If an application needs real-time results, it is better to serve the model closer to where it is needed, such as an edge or IoT device.

Know that edge computing is the practice of moving compute and storage resources closer to the location at which they are needed. Edge computing devices can be relatively simple IoT devices, such as sensors with a small amount of memory and limited processing power. This type of device could be useful when the data processing load is light. Edge computing is used when low-latency data processing is needed—for example, to control machinery such as autonomous vehicles or manufacturing equipment. To enable edge computing, the system architecture has to be designed to provide compute, storage, and networking capabilities at the edge while services run in the cloud or in an on-premises data center for the centralized management of devices and centrally stored data.

Be able to list the three basic components of edge computing. Edge computing consists of edge devices, gateway devices, and the cloud platform. Edge devices provide three kinds of data: metadata about the device, state information about the device, and telemetry data. Before a device is incorporated into an IoT processing system, it must be provisioned. After a device is provisioned and it starts collecting data, the data is then processed on the device. After local processing, data is transmitted to a gateway. Gateways can manage network traffic across protocols. Data sent to the cloud is ingested by one of a few different kinds of services in GCP, including Cloud Pub/Sub, IoT Core MQTT, and Stackdriver Monitoring and Logging.

Know that an Edge TPU is a hardware device available from Google for implementing edge computing. This device is an application-specific integrated circuit (ASIC) designed for running AI services at the edge. Edge TPU is designed to work with Cloud TPU and Google Cloud services. In addition to the hardware, Edge TPU includes software and AI algorithms.

Know that Cloud IoT is Google's managed service for IoT services. This platform provides services for integrating edge computing with centralized processing services. Device data is captured by the Cloud IoT Core service, which can then publish data to Cloud Pub/Sub for streaming analytics. Data can also be stored in BigQuery for analysis or used for training new machine learning models in Cloud ML. Data provided through Cloud IoT can also be used to trigger Cloud Functions and associated workflows.

Understand GPUs and TPUs. Graphic processing units are accelerators that have multiple arithmetic logic units (ALUs) that implement adders and multipliers. This architecture is well suited to workloads that benefit from massive parallelization, such as training deep learning models. GPUs and CPUs are both subject to the von Neumann bottleneck, which is the limited data rate between a processor and memory, and slow processing. TPUs are specialized accelerators based on ASICs and created by Google to improve training of deep neural networks. These accelerators are designed for the TensorFlow framework. TPUs reduces the impact of the von Neumann bottleneck by implementing matrix multiplication in the processor. Know the criteria for choosing between CPUs, GPUs, and TPUs.

Review Questions

You can find the answers in the appendix.

1. You are in the early stages of developing a machine learning model using a framework that requires high-precision arithmetic and benefits from massive parallelization. Your data set fits within 32 GB of memory. You want to use Jupyter Notebooks to build the model iteratively and analyze results. What kind of infrastructure would you use?

1. A TPU pod with at least 32 TPUs
2. A single TPU
3. A single server with a GPU
4. A managed instance group of 4 VMs with 64 GB of memory each

2. You are developing a machine learning model that will predict failures in high-precision machining equipment. The equipment has hundreds of IoT sensors that send telemetry data every second. Thousands of the machines are in use in a variety of operating conditions. A year's worth of data is available for model training. You plan to use TensorFlow, a synchronous training strategy, and TPUs. Which of the following strategies would you use?

1. MirroredStrategy
2. CentralStorageStrategy
3. MultiWorkerMirroredStrategy
4. TPUStrategy

3. Your client has developed a machine learning model that detects anomalies in equity trading time-series data. The model runs as a service in a Google Kubernetes Engine (GKE) cluster deployed in the us-west-1 region. A number of financial institutions in New York and London are interested in licensing the technology, but they are concerned that the total time required to make a prediction is longer than they can tolerate. The distance between the serving infrastructure and New York is about

4,800 kilometers, and the distance to London is about 8,000 kilometers. This is an example of what kind of problem with serving a machine learning model?

1. Overfitting
2. Underfitting
3. Latency
4. Scalability

4. A study of global climate change is building a network of environmental sensors distributed across the globe. Sensors are deployed in groups of 12 sensors and a gateway. An analytics pipeline is implemented in GCP. Data will be ingested by Cloud Pub/Sub and analyzed using the stream processing capabilities of Cloud Dataflow. The analyzed data will be stored in BigQuery for further analysis by scientists. The bandwidth between the gateways and the GCP is limited and sometimes unreliable. The scientists have determined that they need the average temperature, pressure, and humidity measurements of each group of 12 sensors for a one-minute period. Each sensor sends data to the gateway every second. This generates 720 data points (12 sensors \times 60 seconds) every minute for each of the three measurements. The scientists only need the one-minute average for temperature, pressure, and humidity. What data processing strategy would you implement?

1. Send all 720 data points for each measurement each minute to a Cloud Pub/Sub message, generate the averages using Cloud Dataflow, and write those results to BigQuery.
2. Average all 720 data points for each measurement each minute, send the average to a Cloud Pub/Sub message, and use Cloud Dataflow and write those results to BigQuery.
3. Send all 720 data points for each measurement each minute to a BigQuery streaming insert into a partitioned table.
4. Average all 720 data points for each measurement each minute, send the average to a Cloud Pub/Sub message, and use Cloud Dataflow and write those results to BigQuery.

5. Your DevOps team is deploying an IoT system to monitor and control environmental conditions in your building. You are using a standard IoT architecture. Which of the following components would you not use?

1. Edge devices
2. Gateways
3. Repeater
4. Cloud platform services

6. In the Google Cloud Platform IoT reference model, which of the following GCP services is used for ingestion?

1. Cloud Storage
2. BigQuery streaming inserts
3. Cloud Pub/Sub
4. Cloud Bigtable

7. A startup is developing a product for autonomous vehicle manufacturers that will enable its vehicles to detect objects better in adverse weather conditions. The product uses a machine learning model built on TensorFlow. Which of the following options would you choose to serve this model?

1. On GKE using TensorFlow Training (TFJob)
2. On Compute Engine using managed instance groups
3. On Edge TPU devices in the vehicles
4. On GPUs in the vehicles

8. In the Google Cloud Platform IoT reference model, which of the following GCP services is used for stream processing?

1. Cloud Storage
2. BigQuery streaming inserts
3. Cloud Pub/Sub
4. Cloud Dataflow

9. You have developed a TensorFlow model using only the most basic TensorFlow operations and no custom operations. You have a large volume of data available for training, but by your estimates it could take several weeks to train the model using a 16 vCPU Compute Engine instance. Which of the following should you try instead?


1. A 32 vCPU Compute Engine instance
2. A TPU pod
3. A GKE cluster using on CPUs

4. App Engine Second Generation

10. You have developed a machine learning model that uses a specialized Fortran library that is optimized for highly parallel, high-precision arithmetic. You only have access to the compiled code and cannot make any changes to source code. You want to use an accelerator to reduce the training time of your model. Which of the following options would you try first?
- 1. A Compute Engine instance with GPUs
 - 2. A TPU pod
 - 3. A Compute Engine instance with CPUs only
 - 4. Cloud Functions

[Support / Sign Out](#)

 **PREV**
[Chapter 9 Deploying Machine Learning Pipelines](#)

NEXT 
[Chapter 11 Measuring, Monitoring, and Troubleshooting Mac...](#)