Module: tf.compat.v1.tpu / tf.tpu

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu#top_of_page)
* [Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu#modules)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu#functions)

Ops related to Tensor Processing Units.

Modules

[experimental](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/experimental) module: Public API for tf.tpu.experimental namespace.

Classes

[class CrossShardOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/CrossShardOptimizer): An optimizer that averages gradients across TPU shards.

Functions

[batch\_parallel(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/batch_parallel): Shards computation along the batch dimension for parallel execution.

[bfloat16\_scope(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/bfloat16_scope): Scope class for bfloat16 variables so that the model uses custom getter.

[core(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/core): Returns the device name for a core in a replicated TPU computation.

[cross\_replica\_sum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/cross_replica_sum): Sum the input tensor across replicas according to group\_assignment.

[initialize\_system(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/initialize_system): Initializes a distributed TPU system for use with TensorFlow.

[outside\_compilation(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/outside_compilation): Builds part of a computation outside any current TPU replicate scope.

[replicate(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/replicate): Builds a graph operator that runs a replicated TPU computation.

[rewrite(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/rewrite): Rewrites computation for execution on a TPU system.

[shard(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/shard): Shards computation for parallel execution.

[shutdown\_system(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/shutdown_system): Shuts down a running a distributed TPU system.

# tf.compat.v1.tpu.batch\_parallel

Shards computation along the batch dimension for parallel execution.

tf.compat.v1.tpu.batch\_parallel(  
    computation,  
    inputs=None,  
    num\_shards=1,  
    infeed\_queue=None,  
    device\_assignment=None,  
    name=None  
)

Defined in [python/tpu/tpu.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/tpu.py).

Convenience wrapper around shard().

inputs must be a list of Tensors or None (equivalent to an empty list). Each input is split into num\_shards pieces along the 0-th dimension, and computation is applied to each shard in parallel.

Tensors are broadcast to all shards if they are lexically captured by computation. e.g.,

x = tf.constant(7) def computation(): return x + 3 ... = shard(computation, ...)

The outputs from all shards are concatenated back together along their 0-th dimension.

Inputs and outputs of the computation must be at least rank-1 Tensors.

#### Args:

* **computation**: A Python function that builds a computation to apply to each shard of the input.
* **inputs**: A list of input tensors or None (equivalent to an empty list). The 0-th dimension of each Tensor must have size divisible by num\_shards.
* **num\_shards**: The number of shards.
* **infeed\_queue**: If not None, the InfeedQueue from which to append a tuple of arguments as inputs to computation.
* **device\_assignment**: If not None, a DeviceAssignment describing the mapping between logical cores in the computation with physical cores in the TPU topology. Uses a default device assignment if None. The DeviceAssignment may be omitted if each shard of the computation uses only one core, and there is either only one shard, or the number of shards is equal to the number of cores in the TPU system.
* **name**: (Deprecated) Does nothing.

#### Returns:

A list of output tensors.

#### Raises:

* **ValueError**: If num\_shards <= 0

# tf.compat.v1.tpu.bfloat16\_scope

Scope class for bfloat16 variables so that the model uses custom getter.

tf.compat.v1.tpu.bfloat16\_scope()

Defined in [python/tpu/bfloat16.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/bfloat16.py).

This enables variables to be read as bfloat16 type when using get\_variable.

# tf.compat.v1.tpu.core

Returns the device name for a core in a replicated TPU computation.

tf.compat.v1.tpu.core(num)

Defined in [python/tpu/tpu.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/tpu.py).

#### Args:

* **num**: the virtual core number within each replica to which operators should be assigned.

#### Returns:

A device name, suitable for passing to tf.device().

# tf.compat.v1.tpu.CrossShardOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/CrossShardOptimizer#top_of_page)
* [Class CrossShardOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/CrossShardOptimizer#class_crossshardoptimizer)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/CrossShardOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/CrossShardOptimizer#methods)
  + [apply\_gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/CrossShardOptimizer#apply_gradients)

## Class CrossShardOptimizer

An optimizer that averages gradients across TPU shards.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

Defined in [python/tpu/tpu\_optimizer.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/tpu_optimizer.py).

## \_\_init\_\_

\_\_init\_\_(  
    opt,  
    reduction=losses.Reduction.MEAN,  
    name='CrossShardOptimizer',  
    group\_assignment=None  
)

Construct a new cross-shard optimizer.

#### Args:

* **opt**: An existing Optimizer to encapsulate.
* **reduction**: The reduction to apply to the shard losses.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "CrossShardOptimizer".
* **group\_assignment**: Optional 2d int32 lists with shape [num\_groups, num\_replicas\_per\_group] which describles how to apply optimizer to subgroups.

#### Raises:

* **ValueError**: If reduction is not a valid cross-shard reduction.

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

Calls tpu\_ops.cross\_replica\_sum() to sum gradient contributions across replicas, and then applies the real optimizer.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizer constructor.

#### Returns:

An Operation that applies the gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If the grads\_and\_vars is malformed.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    \*\*kwargs  
)

Compute gradients of "loss" for the variables in "var\_list".

This simply wraps the compute\_gradients() from the real optimizer. The gradients will be aggregated in the apply\_gradients() so that user can modify the gradients like clipping with per replica global norm if needed. The global norm with aggregated gradients can be bad as one replica's huge gradients can hurt the gradients from other replicas.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKey.TRAINABLE\_VARIABLES.
* **\*\*kwargs**: Keyword arguments for compute\_gradients().

#### Returns:

A list of (gradient, variable) pairs.

#### Raises:

* **ValueError**: If not within a tpu\_shard\_context or group\_assignment is invalid.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    \*args,  
    \*\*kwargs  
)

Return a slot named "name" created for "var" by the Optimizer.

This simply wraps the get\_slot() from the actual optimizer.

#### Args:

* **\*args**: Arguments for get\_slot().
* **\*\*kwargs**: Keyword arguments for get\_slot().

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names(  
    \*args,  
    \*\*kwargs  
)

Return a list of the names of slots created by the Optimizer.

This simply wraps the get\_slot\_names() from the actual optimizer.

#### Args:

* **\*args**: Arguments for get\_slot().
* **\*\*kwargs**: Keyword arguments for get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

Forwarding the variables from the underlying optimizer.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.tpu.cross\_replica\_sum

Sum the input tensor across replicas according to group\_assignment.

tf.compat.v1.tpu.cross\_replica\_sum(  
    x,  
    group\_assignment=None,  
    name=None  
)

Defined in [python/tpu/ops/tpu\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/ops/tpu_ops.py).

#### Args:

* **x**: The local tensor to the sum.
* **group\_assignment**: Optional 2d int32 lists with shape [num\_groups, num\_replicas\_per\_group]. group\_assignment[i] represents the replica ids in the ith subgroup.
* **name**: Optional op name.

#### Returns:

A Tensor which is summed across replicas.

# tf.compat.v1.tpu.initialize\_system

Initializes a distributed TPU system for use with TensorFlow.

tf.compat.v1.tpu.initialize\_system(  
    embedding\_config=None,  
    job=None  
)

Defined in [python/tpu/tpu.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/tpu.py).

#### Args:

* **embedding\_config**: If not None, a TPUEmbeddingConfiguration proto describing the desired configuration of the hardware embedding lookup tables. If embedding\_config is None, no hardware embeddings can be used.
* **job**: The job (the XXX in TensorFlow device specification /job:XXX) that contains the TPU devices that will be initialized. If job=None it is assumed there is only one job in the TensorFlow flock, and an error will be returned if this assumption does not hold.

#### Returns:

A serialized TopologyProto that describes the TPU system. Note: the topology must be evaluated using Session.run before it can be used.

# tf.compat.v1.tpu.outside\_compilation

Builds part of a computation outside any current TPU replicate scope.

tf.compat.v1.tpu.outside\_compilation(  
    computation,  
    \*args,  
    \*\*kwargs  
)

Defined in [python/tpu/tpu.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/tpu.py).

#### Args:

* **computation**: A Python function that builds the computation to place on the host.
* **\*args**: the positional arguments for the computation.
* **\*\*kwargs**: the keyword arguments for the computation.

#### Returns:

The Tensors returned by computation.

# tf.compat.v1.tpu.replicate

Builds a graph operator that runs a replicated TPU computation.

tf.compat.v1.tpu.replicate(  
    computation,  
    inputs=None,  
    infeed\_queue=None,  
    device\_assignment=None,  
    name=None,  
    maximum\_shapes=None  
)

Defined in [python/tpu/tpu.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/tpu.py).

#### Args:

* **computation**: A Python function that builds the computation to replicate.
* **inputs**: A list of lists of input tensors or None (equivalent to [[]]), indexed by [replica\_num][input\_num]. All replicas must have the same number of inputs. Each input can be a nested structure containing values that are convertible to tensors. Note that passing an N-dimension list of compatible values will result in a N-dimention list of scalar tensors rather than a single Rank-N tensors. If you need different behavior, convert part of inputs to tensors with [tf.convert\_to\_tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/convert_to_tensor).
* **infeed\_queue**: If not None, the InfeedQueue from which to append a tuple of arguments as inputs to computation.
* **device\_assignment**: If not None, a DeviceAssignment describing the mapping between logical cores in the computation with physical cores in the TPU topology. Uses a default device assignment if None. The DeviceAssignment may be omitted if each replica of the computation uses only one core, and there is either only one replica, or the number of replicas is equal to the number of cores in the TPU system.
* **name**: (Deprecated) Does nothing.
* **maximum\_shapes**: A nested structure of tf.TensorShape representing the shape to which the respective component of each input element in each replica should be padded. Any unknown dimensions (e.g. tf.compat.v1.Dimension(None) in a tf.TensorShape or -1 in a tensor-like object) will be padded to the maximum size of that dimension over all replicas. Note that if the input dimension is already static, we won't do padding on it and we require the maximum\_shapes to have the same value or None on that dimension. The structure of maximum\_shapes needs to be the same as inputs[0].

#### Returns:

A list of outputs, indexed by [replica\_num] each output can be a nested structure same as what computation() returns with a few exceptions.

Exceptions include: 1) None output: a NoOp would be returned which control-depends on computation. 2) Single value output: A tuple containing the value would be returned. 3) Operation-only outputs: a NoOp would be returned which control-depends on computation. TODO(b/121383831): Investigate into removing these special cases.

#### Raises:

* **ValueError**: If all replicas do not have equal numbers of input tensors.
* **ValueError**: If the number of inputs per replica does not match the number of formal parameters to computation.
* **ValueError**: If the static inputs dimensions don't match with the values given in maximum\_shapes.
* **ValueError**: If the structure of inputs per replica does not match the structure of maximum\_shapes.

# tf.compat.v1.tpu.rewrite

Rewrites computation for execution on a TPU system.

tf.compat.v1.tpu.rewrite(  
    computation,  
    inputs=None,  
    infeed\_queue=None,  
    device\_assignment=None,  
    name=None  
)

Defined in [python/tpu/tpu.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/tpu.py).

#### Args:

* **computation**: A Python function that builds a computation to apply to the input. If the function takes n inputs, 'inputs' should be a list of n tensors.

computation may return a list of operations and tensors. Tensors must come before operations in the returned list. The return value of rewrite is a list of tensors corresponding to the tensors from the output of computation.

All Operations constructed during computation will be executed when evaluating any of the returned output tensors, not just the ones returned.

* **inputs**: A list of input tensors or None (equivalent to an empty list). Each input can be a nested structure containing values that are convertible to tensors. Note that passing an N-dimension list of compatible values will result in a N-dimention list of scalar tensors rather than a single Rank-N tensors. If you need different behavior, convert part of inputs to tensors with [tf.convert\_to\_tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/convert_to_tensor).
* **infeed\_queue**: If not None, the InfeedQueue from which to append a tuple of arguments as inputs to computation.
* **device\_assignment**: if not None, a DeviceAssignment describing the mapping between logical cores in the computation with physical cores in the TPU topology. May be omitted for a single-core computation, in which case the core attached to task 0, TPU device 0 is used.
* **name**: (Deprecated) Does nothing.

#### Returns:

Same data structure as if computation(\*inputs) is called directly with some exceptions for correctness. Exceptions include: 1) None output: a NoOp would be returned which control-depends on computation. 2) Single value output: A tuple containing the value would be returned. 3) Operation-only outputs: a NoOp would be returned which control-depends on computation. TODO(b/121383831): Investigate into removing these special cases.

# tf.compat.v1.tpu.shard

Shards computation for parallel execution.

tf.compat.v1.tpu.shard(  
    computation,  
    inputs=None,  
    num\_shards=1,  
    input\_shard\_axes=None,  
    outputs\_from\_all\_shards=True,  
    output\_shard\_axes=None,  
    infeed\_queue=None,  
    device\_assignment=None,  
    name=None  
)

Defined in [python/tpu/tpu.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/tpu.py).

inputs must be a list of Tensors or None (equivalent to an empty list), each of which has a corresponding split axis (from input\_shard\_axes). Each input is split into num\_shards pieces along the corresponding axis, and computation is applied to each shard in parallel.

Tensors are broadcast to all shards if they are lexically captured by computation. e.g.,

x = tf.constant(7) def computation(): return x + 3 ... = shard(computation, ...)

TODO(phawkins): consider adding support for broadcasting Tensors passed as inputs.

If outputs\_from\_all\_shards is true, the outputs from all shards of computation are concatenated back together along their output\_shards\_axes. Otherwise, each output is taken from an arbitrary shard.

Inputs and outputs of the computation must be at least rank-1 Tensors.

#### Args:

* **computation**: A Python function that builds a computation to apply to each shard of the input.
* **inputs**: A list of input tensors or None (equivalent to an empty list). Each input tensor has a corresponding shard axes, given by input\_shard\_axes, which must have size divisible by num\_shards.
* **num\_shards**: The number of shards.
* **input\_shard\_axes**: A list of dimensions along which to shard inputs, or None. None means "shard all inputs along dimension 0". If not None, there must be one dimension per input.
* **outputs\_from\_all\_shards**: Boolean or list of boolean. For each output, if True, outputs from all shards are concatenated along the corresponding output\_shard\_axes entry. Otherwise, each output is taken from an arbitrary shard. If the argument is a boolean, the argument's value is used for each output.
* **output\_shard\_axes**: A list of dimensions along which to concatenate the outputs of computation, or None. None means "concatenate all outputs along dimension 0". If not None, there must be one dimension per output. Ignored if outputs\_from\_all\_shards is False.
* **infeed\_queue**: If not None, the InfeedQueue to use to augment the inputs of computation.
* **device\_assignment**: If not None, a DeviceAssignment describing the mapping between logical cores in the computation with physical cores in the TPU topology. Uses a default device assignment if None. The DeviceAssignment may be omitted if each shard of the computation uses only one core, and there is either only one shard, or the number of shards is equal to the number of cores in the TPU system.
* **name**: (Deprecated) Does nothing.

#### Returns:

A list of output tensors.

#### Raises:

* **ValueError**: If num\_shards <= 0
* **ValueError**: If len(input\_shard\_axes) != len(inputs)
* **ValueError**: If len(output\_shard\_axes) != len(outputs from computation)

# tf.compat.v1.tpu.shutdown\_system

Shuts down a running a distributed TPU system.

tf.compat.v1.tpu.shutdown\_system(job=None)

Defined in [python/tpu/tpu.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/tpu.py).

#### Args:

* **job**: The job (the XXX in TensorFlow device specification /job:XXX) that contains the TPU devices that will be shutdown. If job=None it is assumed there is only one job in the TensorFlow flock, and an error will be returned if this assumption does not hold.

Module: tf.tpu.experimental

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental#top_of_page)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental#functions)

Public API for tf.tpu.experimental namespace.

Classes

[class DeviceAssignment](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment): Mapping from logical cores in a computation to the physical TPU topology.

Functions

[initialize\_tpu\_system(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/initialize_tpu_system): Initialize the TPU devices.

# tf.tpu.experimental.DeviceAssignment

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment#top_of_page)
* [Class DeviceAssignment](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment#class_deviceassignment)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment#properties)

## Class DeviceAssignment

Mapping from logical cores in a computation to the physical TPU topology.

### Aliases:

* Class tf.compat.v1.tpu.experimental.DeviceAssignment
* Class tf.compat.v2.tpu.experimental.DeviceAssignment
* Class tf.tpu.experimental.DeviceAssignment

Defined in [python/tpu/device\_assignment.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/device_assignment.py).

Prefer to use the DeviceAssignment.build() helper to construct a DeviceAssignment; it is easier if less flexible than constructing a DeviceAssignment directly.

## \_\_init\_\_

\_\_init\_\_(  
    topology,  
    core\_assignment  
)

Constructs a DeviceAssignment object.

#### Args:

* **topology**: A Topology object that describes the physical TPU topology.
* **core\_assignment**: A logical to physical core mapping, represented as a rank 3 numpy array. See the description of the core\_assignment property for more details.

#### Raises:

* **ValueError**: If topology is not Topology object.
* **ValueError**: If core\_assignment is not a rank 3 numpy array.

## Properties

### core\_assignment

The logical to physical core mapping.

#### Returns:

An integer numpy array of rank 3, with shape [num\_replicas, num\_cores\_per\_replica, topology\_rank]. Maps (replica, logical core) pairs to physical topology coordinates.

### num\_cores\_per\_replica

The number of cores per replica.

### num\_replicas

The number of replicas of the computation.

### topology

A Topology that describes the TPU topology.

## Methods

### build

@staticmethod  
build(  
    topology,  
    computation\_shape=None,  
    computation\_stride=None,  
    num\_replicas=1  
)

### coordinates

coordinates(  
    replica,  
    logical\_core  
)

Returns the physical topology coordinates of a logical core.

### host\_device

host\_device(  
    replica=0,  
    logical\_core=0,  
    job=None  
)

Returns the CPU device attached to a logical core.

### lookup\_replicas

lookup\_replicas(  
    task\_id,  
    logical\_core  
)

Lookup replica ids by task number and logical core.

#### Args:

* **task\_id**: TensorFlow task number.
* **logical\_core**: An integer, identifying a logical core.

#### Returns:

A sorted list of the replicas that are attached to that task and logical\_core.

#### Raises:

* **ValueError**: If no replica exists in the task which contains the logical core.

### tpu\_device

tpu\_device(  
    replica=0,  
    logical\_core=0,  
    job=None  
)

Returns the name of the TPU device assigned to a logical core.

### tpu\_ordinal

tpu\_ordinal(  
    replica=0,  
    logical\_core=0  
)

Returns the ordinal of the TPU device assigned to a logical core.

# tf.tpu.experimental.initialize\_tpu\_system

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/initialize_tpu_system#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/initialize_tpu_system#aliases)

Initialize the TPU devices.

### Aliases:

* tf.compat.v1.tpu.experimental.initialize\_tpu\_system
* tf.compat.v2.tpu.experimental.initialize\_tpu\_system
* tf.tpu.experimental.initialize\_tpu\_system

tf.tpu.experimental.initialize\_tpu\_system(cluster\_resolver=None)

Defined in [python/tpu/tpu\_strategy\_util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/tpu_strategy_util.py).

#### Args:

* **cluster\_resolver**: A tf.distribute.cluster\_resolver.TPUClusterResolver, which provides information about the TPU cluster.

#### Returns:

The tf.tpu.Topology object for the topology of the TPU cluster.

#### Raises:

* **RuntimeError**: If no TPU devices found for eager execution.