tf.estimator.RunConfig

Contents

Class **RunConfig**

Defined in `tensorflow/python/estimator/run_config.py`.

This class specifies the configurations for an `Estimator` run.

Properties

`cluster_spec`

`evaluation_master`

`is_chief`

`keep_checkpoint_every_n_hours`

`keep_checkpoint_max`

`log_step_count_steps`

`master`

`model_dir`

`num_ps_replicas`

`num_worker_replicas`

`save_checkpoints_secs`

`save_checkpoints_steps`

`save_summary_steps`

`service`

Returns the platform defined (in TF_CONFIG) service dict.

session_config

task_id

task_type

tf_random_seed

Methods

__init__

```
__init__(
    model_dir=None,
    tf_random_seed=None,
    save_summary_steps=100,
    save_checkpoints_steps=_USE_DEFAULT,
    save_checkpoints_secs=_USE_DEFAULT,
    session_config=None,
    keep_checkpoint_max=5,
    keep_checkpoint_every_n_hours=10000,
    log_step_count_steps=100
)
```

Constructs a RunConfig.

All distributed training related properties `cluster_spec`, `is_chief`, `master`, `num_worker_replicas`, `num_ps_replicas`, `task_id`, and `task_type` are set based on the `TF_CONFIG` environment variable, if the pertinent information is present. The `TF_CONFIG` environment variable is a JSON object with attributes: `cluster` and `task`.

`cluster` is a JSON serialized version of `ClusterSpec`'s Python dict from `server_lib.py`, mapping task types (usually one of the `TaskType` enums) to a list of task addresses.

`task` has two attributes: `type` and `index`, where `type` can be any of the task types in `cluster`. `When` TF_CONFIG` contains said information, the following properties are set on this class:

- `cluster_spec` is parsed from `TF_CONFIG['cluster']`. Defaults to {}. If present, must have one and only one node in the `chief` attribute of `cluster_spec`.
- `task_type` is set to `TF_CONFIG['task']['type']`. Must set if `cluster_spec` is present; must be `worker` (the default value) if `cluster_spec` is not set.
- `task_id` is set to `TF_CONFIG['task']['index']`. Must set if `cluster_spec` is present; must be 0 (the default value) if `cluster_spec` is not set.
- `master` is determined by looking up `task_type` and `task_id` in the `cluster_spec`. Defaults to ''.
- `num_ps_replicas` is set by counting the number of nodes listed in the `ps` attribute of `cluster_spec`. Defaults to 0.
- `num_worker_replicas` is set by counting the number of nodes listed in the `worker` and `chief` attributes of `cluster_spec`. Defaults to 1.
- `is_chief` is determined based on `task_type` and `cluster`.

There is a special node with `task_type` as `evaluator`, which is not part of the (training) `cluster_spec`. It handles the distributed evaluation job.

Example of non-chief node:

```
cluster = {'chief': ['host0:2222'],
           'ps': ['host1:2222', 'host2:2222'],
           'worker': ['host3:2222', 'host4:2222', 'host5:2222']}
os.environ['TF_CONFIG'] = json.dumps(
    {'cluster': cluster,
     'task': {'type': 'worker', 'index': 1}})
config = ClusterConfig()
assert config.master == 'host4:2222'
assert config.task_id == 1
assert config.num_ps_replicas == 2
assert config.num_worker_replicas == 4
assert config.cluster_spec == server_lib.ClusterSpec(cluster)
assert config.task_type == 'worker'
assert not config.is_chief
```

Example of chief node:

```
cluster = {'chief': ['host0:2222'],
           'ps': ['host1:2222', 'host2:2222'],
           'worker': ['host3:2222', 'host4:2222', 'host5:2222']}
os.environ['TF_CONFIG'] = json.dumps(
    {'cluster': cluster,
     'task': {'type': 'chief', 'index': 0}})
config = ClusterConfig()
assert config.master == 'host0:2222'
assert config.task_id == 0
assert config.num_ps_replicas == 2
assert config.num_worker_replicas == 4
assert config.cluster_spec == server_lib.ClusterSpec(cluster)
assert config.task_type == 'chief'
assert config.is_chief
```

Example of evaluator node (evaluator is not part of training cluster):

```
cluster = {'chief': ['host0:2222'],
           'ps': ['host1:2222', 'host2:2222'],
           'worker': ['host3:2222', 'host4:2222', 'host5:2222']}
os.environ['TF_CONFIG'] = json.dumps(
    {'cluster': cluster,
     'task': {'type': 'evaluator', 'index': 0}})
config = ClusterConfig()
assert config.master == ''
assert config.evaluator_master == ''
assert config.task_id == 0
assert config.num_ps_replicas == 0
assert config.num_worker_replicas == 0
assert config.cluster_spec == {}
assert config.task_type == 'evaluator'
assert not config.is_chief
```

N.B.: If `save_checkpoints_steps` or `save_checkpoints_secs` is set, `keep_checkpoint_max` might need to be adjusted accordingly, especially in distributed training. For example, setting `save_checkpoints_secs` as 60 without adjusting `keep_checkpoint_max` (defaults to 5) leads to situation that checkpoint would be garbage collected after 5 minutes. In distributed training, the evaluation job starts asynchronously and might fail to load or find the checkpoint due to race condition.

Args:

- `model_dir`: directory where model parameters, graph, etc are saved. If `None`, will use a default value set by the Estimator.

- `tf_random_seed`: Random seed for TensorFlow initializers. Setting this value allows consistency between reruns.
- `save_summary_steps`: Save summaries every this many steps.
- `save_checkpoints_steps`: Save checkpoints every this many steps. Can not be specified with **`save_checkpoints_secs`**.
- `save_checkpoints_secs`: Save checkpoints every this many seconds. Can not be specified with **`save_checkpoints_steps`**. Defaults to 600 seconds if both **`save_checkpoints_steps`** and **`save_checkpoints_secs`** are not set in constructor. If both **`save_checkpoints_steps`** and **`save_checkpoints_secs`** are None, then checkpoints are disabled.
- `session_config`: a ConfigProto used to set session parameters, or None.
- `keep_checkpoint_max`: The maximum number of recent checkpoint files to keep. As new files are created, older files are deleted. If None or 0, all checkpoint files are kept. Defaults to 5 (that is, the 5 most recent checkpoint files are kept.)
- `keep_checkpoint_every_n_hours`: Number of hours between each checkpoint to be saved. The default value of 10,000 hours effectively disables the feature.
- `log_step_count_steps`: The frequency, in number of global steps, that the global step/sec will be logged during training.

Raises:

- `ValueError`: If both **`save_checkpoints_steps`** and **`save_checkpoints_secs`** are set.

`replace`

```
replace(**kwargs)
```

Returns a new instance of **`RunConfig`** replacing specified properties.

Only the properties in the following list are allowed to be replaced: - **`model_dir`**. - **`tf_random_seed`**, - **`save_summary_steps`**, - **`save_checkpoints_steps`**, - **`save_checkpoints_secs`**, - **`session_config`**, - **`keep_checkpoint_max`**, - **`keep_checkpoint_every_n_hours`**, - **`log_step_count_steps`**,

In addition, either **`save_checkpoints_steps`** or **`save_checkpoints_secs`** can be set (should not be both).

Args:

- `**kwargs`: keyword named properties with new values.

Raises:

- `ValueError`: If any property name in **`kwargs`** does not exist or is not allowed to be replaced, or both **`save_checkpoints_steps`** and **`save_checkpoints_secs`** are set.

Returns:

a new instance of **`RunConfig`**.

Stay Connected

Blog

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow

English

Terms | **Privacy**