tf.train.SessionManager

Contents

Class `SessionManager`

Defined in `tensorflow/python/training/session_manager.py`.

See the guide: Training > Distributed execution

Training helper that restores from checkpoint and creates session.

This class is a small wrapper that takes care of session creation and checkpoint recovery. It also provides functions that to facilitate coordination among multiple training threads or processes.

- Checkpointing trained variables as the training progresses.
- Initializing variables on startup, restoring them from the most recent checkpoint after a crash, or wait for checkpoints to become available.

Usage:

```
with tf.Graph().as_default():
   ...add operations to the graph...
  # Create a SessionManager that will checkpoint the model in '/tmp/mydir'.
  sm = SessionManager()
  sess = sm.prepare_session(master, init_op, saver, checkpoint_dir)
  # Use the session to train the graph.
  while True:
    sess.run(<my_train_op>)
```

`prepare_session()` initializes or restores a model. It requires `init_op` and `saver` as an argument.

A second process could wait for the model to be ready by doing the following:

```
with tf.Graph().as_default():
   ...add operations to the graph...
  # Create a SessionManager that will wait for the model to become ready.
  sm = SessionManager()
  sess = sm.wait_for_session(master)
  # Use the session to train the graph.
  while True:
    sess.run(<my_train_op>)
```

`wait_for_session()` waits for a model to be initialized by other processes.

Methods

`__init__`

```
__init__(
    local_init_op=None,
    ready_op=None,
    ready_for_local_init_op=None,
    graph=None,
    recovery_wait_secs=30
)
```

Creates a SessionManager.

The `local_init_op` is an `Operation` that is run always after a new session was created. If `None`, this step is skipped.

The `ready_op` is an `Operation` used to check if the model is ready. The model is considered ready if that operation returns an empty 1D string tensor. If the operation returns a non empty 1D string tensor, the elements are concatenated and used to indicate to the user why the model is not ready.

The `ready_for_local_init_op` is an `Operation` used to check if the model is ready to run local_init_op. The model is considered ready if that operation returns an empty 1D string tensor. If the operation returns a non empty 1D string tensor, the elements are concatenated and used to indicate to the user why the model is not ready.

If `ready_op` is `None`, the model is not checked for readiness.

`recovery_wait_secs` is the number of seconds between checks that the model is ready. It is used by processes to wait for a model to be initialized or restored. Defaults to 30 seconds.

Args:

- `local_init_op`: An `Operation` run immediately after session creation. Usually used to initialize tables and local variables.
- `ready_op`: An `Operation` to check if the model is initialized.
- `ready_for_local_init_op`: An `Operation` to check if the model is ready to run local_init_op.
- `graph`: The `Graph` that the model will use.
- `recovery_wait_secs`: Seconds between checks for the model to be ready.

Raises:

- `ValueError`: If ready_for_local_init_op is not None but local_init_op is None

`prepare_session`

```
prepare_session(
    master,
    init_op=None,
    saver=None,
    checkpoint_dir=None,
    checkpoint_filename_with_path=None,
    wait_for_checkpoint=False,
    max_wait_secs=7200,
    config=None,
    init_feed_dict=None,
    init_fn=None
)
```

Creates a `Session`. Makes sure the model is ready to be used.

Creates a `Session` on 'master'. If a `saver` object is passed in, and `checkpoint_dir` points to a directory containing valid checkpoint files, then it will try to recover the model from checkpoint. If no checkpoint files are available, and `wait_for_checkpoint` is `True`, then the process would check every `recovery_wait_secs`, up to `max_wait_secs`, for recovery to succeed.

If the model cannot be recovered successfully then it is initialized by either running the provided `init_op`, or calling the provided `init_fn`. The local_init_op is also run after init_op and init_fn, regardless of whether the model was recovered successfully, but only if ready_for_local_init_op passes.

It is an error if the model cannot be recovered and no `init_op` or `init_fn` or `local_init_op` are passed.

Args:

- `master`: `String` representation of the TensorFlow master to use.
- `init_op`: Optional `Operation` used to initialize the model.
- `saver`: A `Saver` object used to restore a model.
- `checkpoint_dir`: Path to the checkpoint files. The latest checkpoint in the dir will be used to restore.
- `checkpoint_filename_with_path`: Full file name path to the checkpoint file.
- `wait_for_checkpoint`: Whether to wait for checkpoint to become available.
- `max_wait_secs`: Maximum time to wait for checkpoints to become available.
- `config`: Optional `ConfigProto` proto used to configure the session.
- `init_feed_dict`: Optional dictionary that maps `Tensor` objects to feed values. This feed dictionary is passed to the session `run()` call when running the init op.
- `init_fn`: Optional callable used to initialize the model. Called after the optional `init_op` is called. The callable must accept one argument, the session being initialized.

Returns:

A `Session` object that can be used to drive the model.

Raises:

- `RuntimeError`: If the model cannot be initialized or recovered.

Raises:

- `ValueError`: If both checkpoint_dir and checkpoint_filename_with_path are set.

recover_session

```
recover_session(
    master,
    saver=None,
    checkpoint_dir=None,
    checkpoint_filename_with_path=None,
    wait_for_checkpoint=False,
    max_wait_secs=7200,
    config=None
)
```

Creates a `Session`, recovering if possible.

Creates a new session on 'master'. If the session is not initialized and can be recovered from a checkpoint, recover it.

Args:

- `master`: `String` representation of the TensorFlow master to use.
- `saver`: A `Saver` object used to restore a model.
- `checkpoint_dir`: Path to the checkpoint files. The latest checkpoint in the dir will be used to restore.
- `checkpoint_filename_with_path`: Full file name path to the checkpoint file.
- `wait_for_checkpoint`: Whether to wait for checkpoint to become available.
- `max_wait_secs`: Maximum time to wait for checkpoints to become available.
- `config`: Optional `ConfigProto` proto used to configure the session.

Returns:

A pair (sess, initialized) where 'initialized' is `True` if the session could be recovered and initialized, `False` otherwise.

Raises:

- `ValueError`: If both checkpoint_dir and checkpoint_filename_with_path are set.

wait_for_session

```
wait_for_session(
    master,
    config=None,
    max_wait_secs=float('Inf')
)
```

Creates a new `Session` and waits for model to be ready.

Creates a new `Session` on 'master'. Waits for the model to be initialized or recovered from a checkpoint. It's expected that another thread or process will make the model ready, and that this is intended to be used by threads/processes that participate in a distributed training configuration where a different thread/process is responsible for initializing or recovering the model being trained.

NB: The amount of time this method waits for the session is bounded by max_wait_secs. By default, this function will wait indefinitely.

Args:

- `master`: `String` representation of the TensorFlow master to use.
- `config`: Optional ConfigProto proto used to configure the session.
- `max_wait_secs`: Maximum time to wait for the session to become available.

Returns:

A `Session`. May be None if the operation exceeds the timeout specified by config.operation_timeout_in_ms.

Raises:

- `tf.DeadlineExceededError`: if the session is not available after max_wait_secs.
