tf.train.MonitoredSession

Contents

Class `MonitoredSession`

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

See the guides: Threading and Queues > Queue usage overview, Training > Distributed execution

Session-like object that handles initialization, recovery and hooks.

Example usage:

```
saver_hook = CheckpointSaverHook(...)
summary_hook = SummarySaverHook(...)
with MonitoredSession(session_creator=ChiefSessionCreator(...),
                      hooks=[saver_hook, summary_hook]) as sess:
  while not sess.should_stop():
    sess.run(train_op)
```

Initialization: At creation time the monitored session does following things in given order:

- calls `hook.begin()` for each given hook
- finalizes the graph via `scaffold.finalize()`
- create session
- initializes the model via initialization ops provided by `Scaffold`
- restores variables if a checkpoint exists
- launches queue runners
- calls `hook.after_create_session()`

Run: When `run()` is called, the monitored session does following things:

- calls `hook.before_run()`
- calls TensorFlow `session.run()` with merged fetches and feed_dict
- calls `hook.after_run()`
- returns result of `session.run()` asked by user
- if `AbortedError` or `UnavailableError` occurs, it recovers or reinitializes the session before executing the run() call again

Exit: At the `close()`, the monitored session does following things in order:

- calls `hook.end()`
- closes the queue runners and the session
- suppresses `OutOfRange` error which indicates that all inputs have been processed if the monitored_session is used as a context

How to set `tf.Session` arguments:

- In most cases you can set session arguments as follows:

```
MonitoredSession(
  session_creator=ChiefSessionCreator(master=..., config=...))
```

- In distributed setting for a non-chief worker, you can use following:

```
MonitoredSession(
  session_creator=WorkerSessionCreator(master=..., config=...))
```

See `MonitoredTrainingSession` for an example usage based on chief or worker.

> ⭐ **Note:** This is not a `tf.Session`. For example, it cannot do following:

- it cannot be set as default session.
- it cannot be sent to saver.save.
- it cannot be sent to tf.train.start_queue_runners.

Args:

- `session_creator` : A factory object to create session. Typically a `ChiefSessionCreator` which is the default one.
- `hooks` : An iterable of `SessionRunHook' objects.

Returns:

A MonitoredSession object.

Properties

graph

The graph that was launched in this session.

Methods

__init__

```
__init__(
    session_creator=None,
    hooks=None,
    stop_grace_period_secs=120
)
```

`__enter__`

```
__enter__()
```

`__exit__`

```
__exit__(
    exception_type,
    exception_value,
    traceback
)
```

close

```
close()
```

run

```
run(
    fetches,
    feed_dict=None,
    options=None,
    run_metadata=None
)
```

Run ops in the monitored session.

This method is completely compatible with the `tf.Session.run()` method.

Args:

- `fetches` : Same as `tf.Session.run()` .
- `feed_dict` : Same as `tf.Session.run()` .
- `options` : Same as `tf.Session.run()` .
- `run_metadata` : Same as `tf.Session.run()` .

Returns:

Same as `tf.Session.run()` .

`should_stop`

```
should_stop()
```

Stay Connected

Blog

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow

English

Terms | **Privacy**