tfdbg.GrpcDebugHook

Contents

Class **GrpcDebugHook**

Inherits From: `SessionRunHook`

Defined in `tensorflow/python/debug/wrappers/hooks.py`.

A hook that streams debugger-related events to any grpc_debug_server.

For example, the debugger data server is a grpc_debug_server. The debugger data server writes debugger-related events it receives via GRPC to logdir. This enables debugging features in Tensorboard such as health pills.

When the arguments of debug_utils.watch_graph changes, strongly consider changing arguments here too so that features are available to tflearn users.

Can be used as a monitor/hook for `tf.train.MonitoredSession` s and `tf.contrib.learn` 's `Estimator` s and `Experiment` s.

Methods

`__init__`

```
__init__(
    grpc_debug_server_addresses,
    watch_fn=None,
    thread_name_filter=None,
    log_usage=True
)
```

Constructs a GrpcDebugHook.

Args:

- `grpc_debug_server_addresses` : (`list` of `str`) A list of the gRPC debug server addresses, in the format of , without the "grpc://" prefix. For example: ["localhost:7000", "192.168.0.2:8000"]
- `watch_fn` : A function that allows for customizing which ops to watch at which specific steps. See doc of `dumping_wrapper.DumpingDebugWrapperSession.__init__` for details.
- `thread_name_filter` : Regular-expression white list for threads on which the wrapper session will be active. See doc of `BaseDebugWrapperSession` for more details.

- `log_usage`: (bool) Whether usage is to be logged.

Raises:

- `ValueError`: if any debugger server addresses start with grpc://.

`after_create_session`

```
after_create_session(
    session,
    coord
)
```

Called when new TensorFlow session is created.

This is called to signal the hooks that a new session has been created. This has two essential differences with the situation in which **`begin`** is called:

- When this is called, the graph is finalized and ops can no longer be added to the graph.
- This method will also be called as a result of recovering a wrapped session, not only at the beginning of the overall session.

Args:

- `session`: A TensorFlow Session that has been created.
- `coord`: A Coordinator object which keeps track of all threads.

`after_run`

```
after_run(
    run_context,
    run_values
)
```

Called after each call to run().

The **`run_values`** argument contains results of requested ops/tensors by **`before_run()`**.

The **`run_context`** argument is the same one send to **`before_run`** call. **`run_context.request_stop()`** can be called to stop the iteration.

If **`session.run()`** raises any exceptions then **`after_run()`** is not called.

Args:

- `run_context`: A **`SessionRunContext`** object.
- `run_values`: A SessionRunValues object.

`before_run`

```
before_run(run_context)
```

Called right before a session is run.

Args:

- `run_context` : A session_run_hook.SessionRunContext. Encapsulates information on the run.

Returns:

A session_run_hook.SessionRunArgs object.

`begin`

```
begin()
```

Called once before using the session.

When called, the default graph is the one that will be launched in the session. The hook can modify the graph by adding new operations to it. After the `begin()` call the graph will be finalized and the other callbacks can not modify the graph anymore. Second call of `begin()` on the same graph, should not change the graph.

`end`

```
end(session)
```

Called at the end of session.

The `session` argument can be used in case the hook wants to run final ops, such as saving a last checkpoint.

If `session.run()` raises exception other than OutOfRangeError or StopIteration then `end()` is not called. Note the difference between `end()` and `after_run()` behavior when `session.run()` raises OutOfRangeError or StopIteration. In that case `end()` is called but `after_run()` is not called.

Args:

- `session` : A TensorFlow Session that will be soon closed.

Last updated November 2, 2017.