tensorflow-tracer Documentation

Release 1.0.1

Sayed Hadi Hashemi

Nov 27, 2018

Contents

TensorFlow Runtime Tracer is a web application to monitor and trace TensorFlow scripts in the runtime on the `op` level.

It starts a web server upon the execution of the script. The web interface keeps track of all the session runs and can trace the execution on demand.

The goal of this tool is to facilitate the process of performance tuning with minimal code changes and insignificant runtime overhead. Both Higher-level (tf.estimator.Estimator) and Low-level (tf.train.MonitoredTrainingSession and co) APIs are supported. It also supports horovod and IBM Distributed Deep Learning (DDL). The tracing session can be saved, reloaded, and distributed effortlessly.

CHAPTER 1

Installation

Use `pip` to install:

```
pip install tensorflow-tracer
```

CHAPTER 2

Quick Start

1. Add `tftracer` to your code:

   Estimator API:

   ```
   from tftracer import TracingServer
   ...

   tracing_server = TracingServer()
   estimator.train(input_fn, hooks=[tracing_server.hook])
   ```

   Low-Level API:

   ```
   from tftracer import TracingServer
   ...
   tracing_server = TracingServer()
   with tf.train.MonitoredTrainingSession(hooks=[tracing_server.hook]):
      ...
   ```

2. Browse to:

   ```
   http://0.0.0.0:9999
   ```

CHAPTER 3

Command line

Tracing sessions can be stored either through the web interface or by calling `tftracer.TracingServer.save_session()`.

To reload a session, run this in the terminal:

```
tftracer filename
```

Then browse to:

```
http://0.0.0.0:9999
```

3.1 Full Usage

```
usage: tftracer [-h] [--port PORT] [--ip IP] session_file

positional arguments:
  session_file  Path to the trace session file

optional arguments:
  -h, --help     show this help message and exit
  --port PORT    To what TCP port web server to listen
  --ip IP        To what IP address web server to listen
```

CHAPTER 4

Examples

Higher-Level API <estimator-example.py> Example of using `tftracer.TracingServer` with TensorFlow `estimator` API.

Low-Level API <monitoredtrainingsession-example.py> Example of using `tftracer.TracingServer` with TensorFlow `MonitoredTrainingSession` API.

Horovod: One Process <horovod-example.py> Example of using `tftracer.TracingServer` with `horovod`. In this example only the one process is being traced.

Horovod: All Processes <horovod-all-example.py> Example of using `tftracer.TracingServer` with `horovod`. In this example all processes are being traced.

Timeline <timeline-example.py> Example of using `tftracer.Timeline` to trace and visualize one `session.run` call without a tracing server.

Load Session <load_session-example.py> Example of saving and loading tracing sessions.

TracingServer Options <options-example.py> Example of setting tracing options.

CHAPTER 5

API Reference

5.1 tftracer.TracingServer

5.2 tftracer.Timeline

CHAPTER 6

Known Bugs/Limitations

- Only Python3 is supported.
- The web interface loads javascript/css libraries remotely (e.g. `vue.js`, `ui-kit`, `jquery`, `jquery-ui`, `Google Roboto`, `awesome-icons`, `...`). Therefore an active internet connection is needed to properly render the interface. The tracing server does not require any remote connection.
- All traces are kept in the memory while tracing server is running.
- Tracing uses `tf.train.SessionRunHook` and is unable to trace auxiliary runs such as `init_op`.
- The tracing capability is limited to what `tf.RunMetadata` offers. For example, CUPTI events are missing when tracing a distributed job.
- HTTPS is not supported.

CHAPTER 7

Frequently Asked Questions

7.1 How to trace/visualize just one session run?

Use `tftracer.Timeline`. for example:

```
from tftracer import Timeline
...
with tf.train.MonitoredTrainingSession() as sess:
   with Timeline() as tl:
    sess.run(fetches, **tl.kwargs)
...
tl.visualize(filename)
```

7.2 Comparision to TensorBoard?

The nature of this project is a short-lived light-weight interactive tracing interface to monitor and trace execution on the `op`-level. In comparison `TensorBoard` is a full-featured tool to inspect the application on many levels:

- `tftracer` does not make any assumption about the dataflow DAG. There is no need to add any additional `op` to the data flow dag (i.e. `tf.summary`) or having a `global step`.
- `tftracer` runs as a thread and lives from the start of the execution and lasts until the end of it. `TensorBoard` runs as a separate process and can outlive the main script.

Index