ONNX RUNTIME AMPERE® AI OPTIMIZER (AIO) Documentation

V.1.1.0

AMPERE COMPUTING

Table of Contents

RELEASE NOTES

v1.1.0:

- Updated to use AIO 0.3.0
    - Misc models speed up

OVERVIEW

The Ampere® AIO inference acceleration engine is fully integrated with the ONNX Runtime framework. ONNX models and ONNX Runtime software written with the ONNX Runtime API can run as-is, without modifications.

ONNX RUNTIME FRAMEWORK

Python is installed with the AIO accelerated ONNX Runtime package containing all dependencies. No additional installation steps are needed.

Versions Compatibility

This release is based on ONNX Runtime 1.10.0. Please refer to ONNX Runtime version compatibility documentation, found at ONNX Runtime and ONNX Versioning Guide, to check the compatibility of models built with older versions of ONNX Runtime.

Python

ONNX Runtime 1.10.0 is built for Python 3.8. Regarding other Python versions, contact your Ampere sales representative. If you are using the software through a third party, contact their customer support team for help. You can also contact the Ampere AI team at ai-support@amperecomputing.com.

AIO CONFIGURATIONS

The AIO inference engine can be configured by a set of environment variables for performance and debugging purposes. They can be set in the command line when running ONNX models (e.g., AIO_NUM_THREADS=16 python run.py) or set in the shell initialization script.

AIO_PROCESS_MODE

This variable controls whether the AIO inference engine is used to run the ONNX model:

- 0: AIO is disabled.
- 1: AIO is enabled (Default).

AIO_CPU_BIND

Enables core binding. If enabled, each AIO thread will bind itself to a single core:

- 0: Core binding disabled.
- 1: Core binding enabled (Default).

AIO_MEM_BIND

Binds memory to NUMA (Non-uniform memory access) node 0. For optimal performance, numactl (https://linux.die.net/man/8/numactl) is preferred. numactl bind will affect both the ONNX Runtime framework and the AIO buffers, while the AIO is unable to affect buffers allocated by the ONNX Runtime framework:

- 0: Membind disabled.
- 1: Membind to node 0 (Default).

AIO_NUMA_CPUS

Select cores that the AIO should bind to (if CPU_BIND is enabled):

- Not set: AIO will use the first N cores of the machine, excluding hyper-threaded machines (Default).
- Set: AIO will try to use N first cores from the list of cores for N threads. The list is in space-separated, 0-based number format. For example, selecting cores 0 to 1: AIO_NUMA_CPUS="0 1".

AIO_NUM_THREADS

Specifies the number of cores that the AIO should use:

- Not set: AIO will use one core (Default).
- "all": AIO will use all cores, as specified by AIO_NUMA_CPUS.
- N: AIO will use N cores.

AIO_DEBUG_MODE

Control verbosity of debug messages:

- 0: No messages
- 1: Errors only
- 2: Basic information, warnings, and errors (Default)
- 3: Most messages
- 4: All messages

QUICKSTART

The following instructions run on Altra/Altra Max Linux machines installed **with Docker**. To initialize the AIO environment run:

```
$ wget -O aio-onnxrt.tar.gz "<your_unique_url>"
$ docker load < aio-onnxrt.tar.gz
$ docker run --privileged=true --rm --name onnxrt-aio --network host -it aio-onnxrt-1.10.0:1.1.0
```

Skip the above steps if running **without a Docker** container.

You can try AIO by either running the Jupyter Notebook examples or Python scripts on the CLI level.

To run the Jupyter Notebook QuickStart examples follow the instructions below:

Set AIO_NUM_THREADS to the requested value first.

```
$ export AIO_NUM_THREADS=16; export OMP_NUM_THREADS=16
$ cd /workspace/aio-examples/
$ bash download_models.sh
$ bash start_notebook.sh
```

If you run the Jupyter Notebook Quickstart on a cloud instance, make sure your machine has port 8080 open and on your local device run:

```
$ ssh -N -L 8080:localhost:8080 -I <ssh_key> your_user@xxx.xxx.xxx.xxx
```

Use a browser to point to the URL printed out by the Jupyter Notebook launcher.

You will find the Jupyter Notebook examples (examples.ipynb) under the /classification and /object detection folders.

The examples run through several inference models, visualize results they produce and present the performance numbers.

To use CLI-level scripts:

Set AIO_NUM_THREADS to the requested value first.

```
$ export AIO_NUM_THREADS=16; export OMP_NUM_THREADS=16
$ cd /workspace/aio-examples/
$ bash download_models.sh
$ pip install torch
```

Go to the directory of choice, e.g.,

```
$ cd classification/resnet_50_v1.5
```

Evaluate the model.

```
$ python3 run.py -m resnet_50_v1.5_fp32.onnx -p fp32
```

Ampere Computing® / 4655 Great America Parkway, Suite 601 / Santa Clara, CA 95054 / www.amperecomputing.com

AMP 2019-0039