hughperkins / tensorflow-cl

OpenCL 1.2 implementation for Tensorflow

9,302 commits **11** branches **16** releases **437** contributors Apache-2.0

hughperkins update build-from-source for llvm4, and coriander name — Latest commit `ca00f71` 3 hours ago

doc	update build-from-source for llvm4, and coriander name	3 hours ago
tensorflow	remove misleading blas line from debuggin earlier	14 hours ago
third_party	update build-from-source for llvm4, and coriander name	3 hours ago
tools	remove unused-local-typedef warnings	6 months ago
travis	tweak python install	6 months ago
util/python	Fix python bath.	8 months ago
.gitignore	merge from origin	12 days ago
.gitmodules	rename compiler to coriander	5 days ago
.travis.yml	tweak name of uploaded tesnorflowpkg	6 months ago
ACKNOWLEDGMENTS	TensorFlow: Improve performance of Alexnet	2 years ago

AUTHORS	update authors	7 months ago
BUILD	Automated rollback of change 131452196	9 months ago
LICENSE	TensorFlow: Initial commit of TensorFlow library.	2 years ago
README.md	update readme for llvm 4	3 hours ago
WORKSPACE	Update the bower versions.	8 months ago
bower.BUILD	Update the bower versions.	8 months ago
configure	fix setting of so_suffix in configure	18 days ago
eigen.BUILD	Optimize Bazel external dependencies	8 months ago
farmhash.BUILD	Merge changes from github.	8 months ago
gif.BUILD	Optimize Bazel external dependencies	8 months ago
gmock.BUILD	Optimize Bazel external dependencies	8 months ago
grpc.BUILD	Optimize Bazel external dependencies	8 months ago
jpeg.BUILD	Optimize Bazel external dependencies	8 months ago
jsoncpp.BUILD	Optimize Bazel external dependencies	8 months ago
linenoise.BUILD	tfprof: "Swiss Army Knife Tool" To Explore Your Model.	8 months ago
nanopb.BUILD	Optimize Bazel external dependencies	8 months ago
png.BUILD	Optimize Bazel external dependencies	8 months ago
pytest.ini	add pytest.ini, never remembered to commit before :-P	6 months ago
six.BUILD	Optimize Bazel external dependencies	8 months ago
zlib.BUILD	Optimize Bazel external dependencies	8 months ago

README.md

Tensorflow-cl

Run Tensorflow on OpenCL™ 1.2 devices

Summary

This repo was created from the original Tensorflow repository at:

- https://github.com/tensorflow/tensorflow

Please see the main repository for full Tensorflow documentation. This readme will only focus on the OpenCL porting aspects of Tensorflow.

Good points of this compared to other OpenCL Tensorflows

- compatible with Mac, since doesnt need SPIR ingestor
- should work theoretically on any OpenCL 1.2 GPU device, without needing SPIR 1.2 extension, or OpenCL 2.0.

What works, what doesnt

Things that are working:

- per-element unary operations
- per-element binary operations
- reductions
- backprop

- BLAS, ie matrix multiplications, using Cedric Nugteren's CLBlast library

Things that arent implemented

- need a random-number generator implementation
  - ideally this would be an implementation of cuRNG, in Coriander, but either way, it's not implemented yet
  - should probalby be sufficient to generate random numbers on the cpu, then copy them to gpu, eg see the analysis at http://stackoverflow.com/questions/9912143/how-to-get-a-random-number-in-opencl/16130111#16130111
- need to activate cudnn within tensorflow-cl. The hard bit is mostly done, ie implementing the convolutions inside Coriander, but need to 'switch it on' here

Test results, on v0.14.0 wheel

test	Intel HD5500, beignet 1.2.1	NVIDIA 940M, driver v367.57
unit tests (`py.test -v`)	pass	pass
linear_regression.py	slow, but works	slow, but works
logistic_regression.py	ok	ok
nearest_neighbor.py	ok (accuracy 0.92)	ok (accuracy 0.92)
multilayer_perceptron.py	missing adam	missing adam
recurrent_network.py	missing adam	missing adam
autoencoder.py	missing rmsprop	

Test results, on v0.16.0 wheel

test	Mac Sierra, using Radeon Pro 450 GPU
unit tests (`py.test -v`)	pass
linear_regression.py	slow, but works
logistic_regression.py	ok
nearest_neighbor.py	ok (accuracy 0.92)
multilayer_perceptron.py	missing random number generator, and slice
recurrent_network.py	missing Adam for matrices, missing random number generator
autoencoder.py	missing random number generator, and gradients for Sigmoid

Test results, on git, as of May 27, 2017

test	Mac Sierra, using Radeon Pro 450 GPU
unit tests (`py.test -v`)	pass
linear_regression.py	slow, but works
logistic_regression.py	ok
nearest_neighbor.py	ok (accuracy 0.92)
autoencoder.py	ok
recurrent_network.py	missing Adam nees random number generator
multilayer_perceptron.py	missing random number generator, and slice

Installation

The environments used for testing/development are:

- Ubuntu 16.04, with:
  - NVIDIA K80, and
- Mac Sierra, with:
  - Intel HD Graphics 530
  - Radeon Pro 450 (using a nice Mac Book Pro 4th generation that my employer ASAPP have provided me with recently :-))

Ubuntu 16.04

You can install from wheel:

- You will need:
  - the tensorflow non-gpu installation pre-requisites,
  - an OpenCL 1.2-enabled GPU, and OpenCL 1.2-enabled drivers
  - python 3
- Simply download https://github.com/hughperkins/tensorflow-cl/releases/download/v0.14.0/tensorflow-0.11.0rc0-py3-none-any.whl , and
- Install using pip:

```
pip install --upgrade tensorflow-0.11.0rc0-py3-none-any.whl
```

Mac Sierra

For Mac Sierra, python 3.6, there is a wheel at https://github.com/hughperkins/tensorflow-cl/releases/tag/v0.16.0

- tested on Mac Sierra, using Radeon Pro 450
- to select the Radeon, given that there's probably an Intel HD530 at gpu index 0, make sure to `export CL_GPUOFFSET=1`, which will select the gpu at index 1, ie the Radeon
- you'll need to install python 3.6, and create a virtualenv from it, activate it
- download the tar file from the link just above, and install by doing:

```
tar -xf tf-v0.16-wheel-mac-sierra-py36.tar
cd tf-v0.16-wheel-mac-sierra-py36
pip install --upgrade tensorflow-0.11.0rc0-py3-none-any.whl
```

Piccie of tests running on Mac Sierra:

```
 [-0.22        0.22        0.14       -0.18000001 -0.41999999  0.47999999
  -0.44       -0.46000001 -0.1         0.28      ]]
expected  [1 3 7]
gpu  [1 3 7]
diff 0
PASSED
tensorflow/stream_executor/cl/test/test_unary_ops.py::test[int32-neg-np.negative(a)] func neg int32
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1083] Creating TensorFlow device (/gpu:0) -> (device: 0, na
me: AMD Radeon Pro 450 Compute Engine, pci bus id: 0000.0000)
original  [[ 4 -4  0  1  2 -1 -1  3  0  1]
 [ 1  4  4 -3  1  4  1  4  0 -1]
 [-2  2  1 -1 -4  4 -4 -4  0  2]]
expected  [[-4  4  0 -1 -2  1  1 -3  0 -1]
 [-1 -4 -4  3 -1 -4 -1 -4  0  1]
 [ 2 -2 -1  1  4 -4  4  4  0 -2]]
gpu  [[-4  4  0 -1 -2  1  1 -3  0 -1]
 [-1 -4 -4  3 -1 -4 -1 -4  0  1]
 [ 2 -2 -1  1  4 -4  4  4  0 -2]]
diff 0
PASSED
tensorflow/stream_executor/cl/test/test_unary_ops.py::test[int32-abs-np.abs(a)] func abs int32
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1083] Creating TensorFlow device (/gpu:0) -> (device: 0, na
me: AMD Radeon Pro 450 Compute Engine, pci bus id: 0000.0000)
original  [[ 4 -4  0  1  2 -1 -1  3  0  1]
 [ 1  4  4 -3  1  4  1  4  0 -1]
 [-2  2  1 -1 -4  4 -4 -4  0  2]]
expected  [[4 4 0 1 2 1 1 3 0 1]
 [1 4 4 3 1 4 1 4 0 1]
 [2 2 1 1 4 4 4 4 0 2]]
gpu  [[4 4 0 1 2 1 1 3 0 1]
 [1 4 4 3 1 4 1 4 0 1]
 [2 2 1 1 4 4 4 4 0 2]]
diff 0
PASSED
tensorflow/stream_executor/cl/test/test_unary_ops.py::test[int32-square-np.square(a)] func square int32
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1083] Creating TensorFlow device (/gpu:0) -> (device: 0, na
me: AMD Radeon Pro 450 Compute Engine, pci bus id: 0000.0000)
original  [[ 4 -4  0  1  2 -1 -1  3  0  1]
 [ 1  4  4 -3  1  4  1  4  0 -1]
 [-2  2  1 -1 -4  4 -4 -4  0  2]]
expected  [[16 16  0  1  4  1  1  9  0  1]
 [ 1 16 16  9  1 16  1 16  0  1]
 [ 4  4  1  1 16 16 16 16  0  4]]
gpu  [[16 16  0  1  4  1  1  9  0  1]
 [ 1 16 16  9  1 16  1 16  0  1]
 [ 4  4  1  1 16 16 16 16  0  4]]
diff 0
PASSED

---------- generated xml file: /Users/hugh2/git-local/tensorflow-blas/test/junit-pytest-report.xml ----------
========================================= short test summary info =========================================
XFAIL tensorflow/stream_executor/cl/test/test_binary_ops.py::test[uint8-div-a / b]
XFAIL tensorflow/stream_executor/cl/test/test_binary_ops.py::test[uint8-mul-a * b]
================================== 67 passed, 2 xfailed in 4.09 seconds ===================================
(env3) ~/git-local/tensorflow-blas (tensorflow-cl|…11△3) $ uname -a
Darwin carrot.local 16.5.0 Darwin Kernel Version 16.5.0: Fri Mar  3 16:52:33 PST 2017; root:xnu-3789.51.2~3/RE
LEASE_X86_64 x86_64
(env3) ~/git-local/tensorflow-blas (tensorflow-cl|…11△3) $
```

Build from source

If you want, you can build from source

Testing

Setup

```
pip install -r tensorflow/stream_executor/cl/test/requirements.txt
```

Run

```
py.test -v
```

Design/architecture

- tensorflow code stays 100% NVIDIA® CUDA™
- Coriander compiles the CUDA code into OpenCL
- Cedric Nugteren's CLBlast provides BLAS (matrix multiplications)

Related projects

DNN Libraries

- OpenCL Torch
- DeepCL

OpenCL middleware

- CLBlast BLAS for OpenCL
- Coriander Compile NVIDIA® CUDA™ apps for OpenCL 1.2
- EasyCL Handles running kernels, passing in arguments etc, on OpenCL

News

- May 27 2017:
  - upgrade LLVM, in Coriander, from 3.8.0 to 4.0.0. Thank you to @iame6162013 for inspiring me to do this
- May 10 2017:
  - test results on Mac Sierra with Radeon Pro 450, using v0.16.0 wheel, now approximately in line with earlier results on Ubuntu, using v0.14.0 wheel
    - https://github.com/hughperkins/tensorflow-cl/releases/tag/v0.16.0
- May 9 2017:
  - Mac build runs ok :-) See the release at Mac build and wheel
  - tested on Mac Sierra, using Radeon Pro 450 GPU
- May 2017:
  - My employer ASAPP have given me use of a nice Mac Book Pro 4th Generation, complete with Radeon Pro 450 GPU :-) I've started looking into getting tensorflow-cl to build/run on it. Actually, it already builds. Just some small(-ish?) teething problems with getting it to run. Watch this space, or post/subscribe into Mac build doesnt run yet issue
- Dec 3:
  - BUILT A MAC WHEEL!!! This is entirely untested. But the wheel is here: https://s3.amazonaws.com/hughperkinstravis/cache/tensorflow-cl/travis/tensorflowpkg.tar.gz (Simply untar it, and `pip install` it)
    - corresponding travis log is at https://travis-ci.org/hughperkins/tensorflow-cl/builds/180917138 and https://travis-ci.org/hughperkins/tensorflow-cl/builds/180410593
    - note that I had to built this in several stages, since it's a 3 hour build, and the logs for this are at https://s3.amazonaws.com/hughperkinstravis/cache/tensorflow-cl/travis/90-c520cc1-log.txt and https://s3.amazonaws.com/hughperkinstravis/cache/tensorflow-cl/travis/91-c55079d-log.txt

        - hmmm, doesnt seem to import yet though. https://travis-ci.org/hughperkins/tensorflow-cl#L2419
- Nov 29:
    - Mac build ran to completion! On Travis. Build output https://travis-ci.org/hughperkins/tensorflow-cl/builds/179727517 Yes, it didnt run, didnt create the wheel. But the `build_pip_package` target built to completion. which is a huge step forward :-) Travis script here: .travis.yml
- Nov 25:
    - release wheel v0.14.0
        - this fixes `argmin`, `argmax`, and `softmax`
        - tons of changes under-the-hood
- Nov 10:
    - released wheel v0.13.0
        - beignet test results fairly solidly match K520 results now
        - fixed the regression on `not_equal` operator
        - removed the spam from memory copy
- Nov 9:
    - fixed unary and binary operators on beignet
    - note that the tools/bazel.rc.templ has changed. Please make sure to copy the new value into tools/bazel.rc, or re-run configure (probably need to do `bazel clean` anyway, so might as well do `./configure`)