PyTorch CI deduplication

Date: May 21, 2019
Author: @Edward Yang

Abstract. We run too many jobs on PyTorch CI in pull requests. It's also expensive to run all of these jobs, and it increases test turnaround time. It also exacerbates the problem of flaky tests, as the more builds you run, the more likely you are to run into a flaky test.

Initial state

Here is the full list of jobs we run, based on CircleCI configuration. I've also added some extra notes about cases where we explicitly match against the job name to toggle behavior (e.g., installing nccl to test against it.)

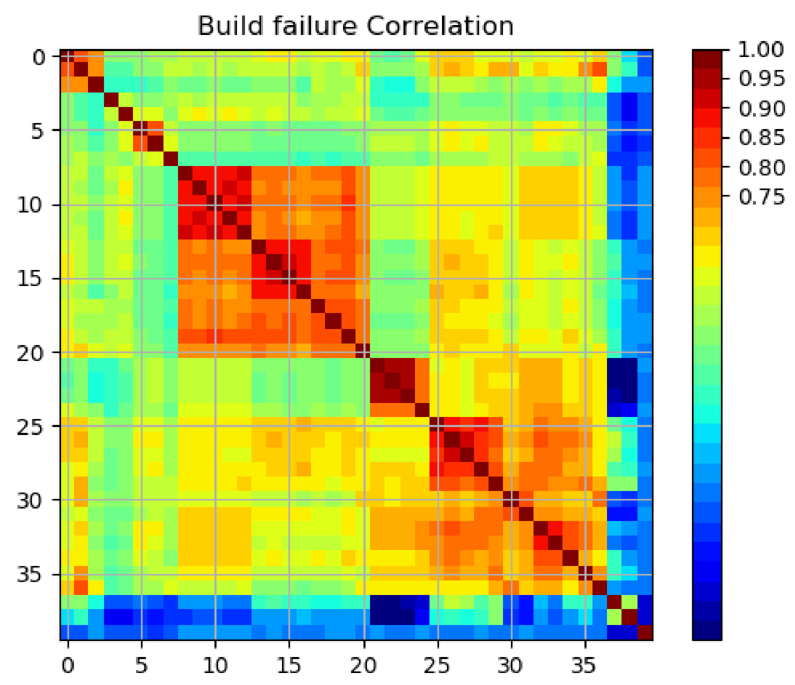
- pytorch-linux-trusty-py2.7.9
    - openmpi
- pytorch-linux-trusty-py2.7
- AAA-pytorch-linux-trusty-py3.5
    - No mkl, mkldnn
- pytorch-linux-trusty-pynightly
- pytorch-linux-trusty-py3.6-gcc4.8
- pytorch-linux-trusty-py3.6-gcc5.4
    - DEBUG=1
- pytorch-**xla**-linux-trusty-py3.6-gcc5.4
- pytorch-**namedtensor**-linux-trusty-py3.6-gcc5.4
- pytorch-linux-trusty-py3.6-gcc7
- pytorch-linux-xenial-py3-clang5-asan
- AAA-pytorch-linux-xenial-cuda9-cudnn7-py2
    - libnccl, openmpi
    - conda cmake
- pytorch-linux-xenial-cuda9-cudnn7-py3 (multigpu, NO_AVX2, NO_AVX-NO_AVX2, slow, nogpu)
    - libnccl, openmpi
    - No mkl, mkldnn
    - docs
    - standalone c10
    - libtorch
    - conda cmake
- pytorch-linux-xenial-cuda9.2-cudnn7-py3-gcc7
    - openmpi
    - libtorch
- pytorch-linux-xenial-cuda10-cudnn7-py3-gcc7
- pytorch-linux-xenial-py3-clang5-android-ndk-r19c
- pytorch_short_perf_test_gpu

- pytorch_doc_push
- pytorch_macos_10_13_py3
- pytorch_macos_10_13_cuda9_2_cudnn7_py3
- caffe2_py2_gcc4_8_ubuntu14_04
- caffe2_py2_gcc4_9_ubuntu14_04
- caffe2_py2_cuda9_0_cudnn7_ubuntu16_04
    - nvidia-machine-learning swizzle
- caffe2_cmake_cuda9_0_cudnn7_ubuntu16_04
- caffe2_py2_cuda9_1_cudnn7_ubuntu16_04
- caffe2_py2_mkl_ubuntu16_04
- caffe2_onnx_py2_gcc5_ubuntu16_04
- caffe2_py2_clang3_8_ubuntu16_04
- caffe2_py2_clang3_9_ubuntu16_04
- caffe2_py2_clang7_ubuntu16_04
- caffe2_py2_android_ubuntu16_04
- caffe2_py2_cuda9_0_cudnn7_centos7
    - glib2
- caffe2_py2_ios_macos10_13_build
- binary_linux_manywheel_2.7mu_cpu_devtoolset3
- binary_linux_manywheel_3.7m_cu100_devtoolset3
- binary_linux_conda_2.7_cpu
- binary_macos_wheel_3.6_cpu
- binary_macos_conda_2.7_cpu
- binary_macos_libtorch_2.7_cpu

Build status correlation

Let's visualize correlation of build statuses based on historical PR build status data. Methodology at ⚓Appendix: Build status correlation methodology.

COMBINED BUILD AND TEST FAILURE CORRELATION



Labels:

```
0. ci/circleci: binary_linux_conda_3.7_cu100
1. ci/circleci: caffe2_py2_ios_macos10_13
2. ci/circleci: binary_macos_conda_3.5_cpu
3. pr/pytorch-win-ws2016-cuda9-cudnn7-py3
4. pr/py2-clang7-rocmdeb-ubuntu16.04
5. pr/caffe2-py2-devtoolset7-rocmrpm-centos7.5
6. pr/caffe2-py2-clang7-rocmdeb-ubuntu16.04
7. pr/caffe2-py2-cuda9.0-cudnn7-windows
8. ci/circleci: pytorch_linux_trusty_py3_5
9. ci/circleci: pytorch_linux_trusty_py2_7_9
10. ci/circleci: pytorch_linux_trusty_py3_6_gcc4_8
11. ci/circleci: pytorch_linux_trusty_py2_7
12. ci/circleci: pytorch_linux_trusty_pynightly
13. ci/circleci: pytorch_linux_xenial_cuda9_cudnn7_py3
14. ci/circleci: pytorch_linux_xenial_cuda9_cudnn7_py2
15. ci/circleci: pytorch_linux_xenial_cuda9_2_cudnn7_py3_gcc7
16. ci/circleci: pytorch_linux_xenial_cuda8_cudnn7_py3
17. ci/circleci: pytorch_linux_xenial_py3_clang5_asan
18. ci/circleci: pytorch_linux_trusty_py3_6_gcc5_4
19. ci/circleci: pytorch_linux_trusty_py3_6_gcc7
20. ci/circleci: pytorch_macos_10_13_py3
21. ci/circleci: caffe2_py2_clang7_ubuntu16_04
22. ci/circleci: caffe2_py2_clang3_8_ubuntu16_04
23. ci/circleci: caffe2_py2_clang3_9_ubuntu16_04
24. ci/circleci: binary_linux_manywheel_2.7mu_cpu
25. ci/circleci: caffe2_py2_cuda9_0_cudnn7_centos7
26. ci/circleci: caffe2_py2_cuda9_1_cudnn7_ubuntu16_04
27. ci/circleci: caffe2_py2_cuda9_0_cudnn7_ubuntu16_04
28. ci/circleci: caffe2_py2_cuda8_0_cudnn7_ubuntu16_04
```

```
29. ci/circleci: caffe2_cmake_cuda9_0_cudnn7_ubuntu16_04
30. ci/circleci: pytorch_macos_10_13_cuda9_2_cudnn7_py3
31. ci/circleci: pytorch_linux_xenial_cuda10_cudnn7_py3_gcc7
32. ci/circleci: caffe2_py2_mkl_ubuntu16_04
33. ci/circleci: caffe2_py2_gcc4_8_ubuntu14_04
34. ci/circleci: caffe2_onnx_py2_gcc5_ubuntu16_04
35. ci/circleci: caffe2_py2_android_ubuntu16_04
36. ci/circleci: caffe2_py2_system_macos10_13
37. ci/circleci: pytorch_short_perf_test_gpu
38. ci/circleci: pytorch_doc_push
39. continuous-integration/travis-ci/pr
```

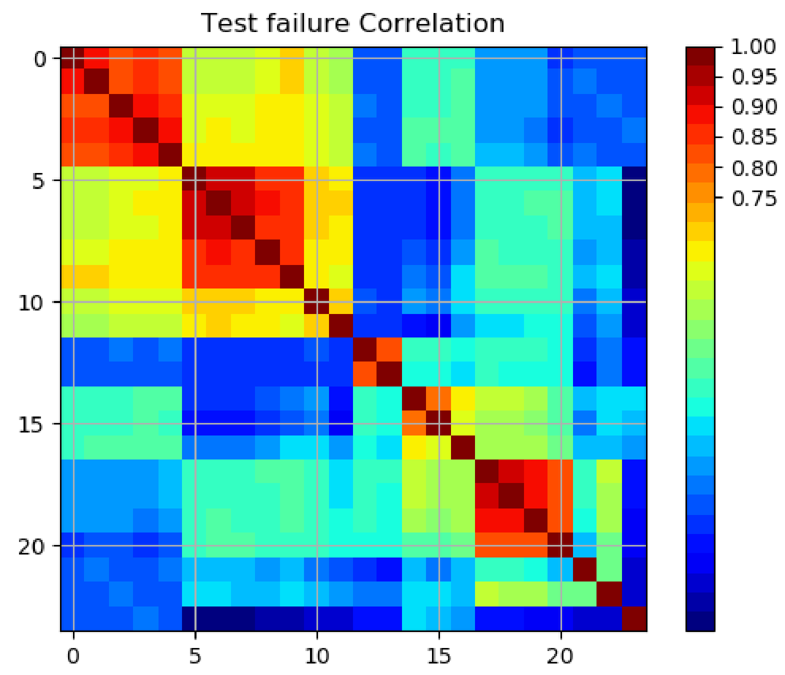
Analysis: Clusterer and visual analysis suggests the following clusters (I only discuss clusters larger than size one)

- 1-3. MacOS/IOS jobs (oddly, binary Linux Conda builds show up here too)
  - Note: the iOS job is actually fairly correlated with OSX; that's what the red square at 1, 36 is about
- 4-6: ROCm
  - 5-6: Caffe2
  - Warning: We're missing the PyTorch CentOS ROCm builds in this data set. A subsequent analysis showed that the PyTorch ROCm builds were also correlated.
- 8-20: PyTorch
  - 8-12: CPU
  - 13-16: CUDA
  - 17-20: Newer compilers (e.g., clang, gcc5 or newer)
- 21-24: Caffe2 clang + manywheel 2.7
  - 21-23: Caffe2 clang
- 25-29: Caffe2 CUDA
- 30-31: Miscellaneous PyTorch jobs (oddly, not very correlated with the primary set)
- 33-36: Miscellaneous Caffe2 jobs (somewhat correlated, but it's very murky)

Surprising things about the clustering: PyTorch CPU and CUDA builds are way more correlated than I thought they would be. On reflection, this is not that surprising, because most build errors likely happen in CPU, but it is a cautionary tale about just looking at correlation (we definitely don't want to remove CUDA builds!)

CORRELATION OF TEST FAILURES ONLY

(Jobs which failed in build are not included.)



Labels:

```
0. ci/circleci: pytorch_linux_trusty_py2_7_test
1. ci/circleci: pytorch_linux_trusty_py2_7_9_test
2. ci/circleci: pytorch_linux_trusty_py3_5_test
3. ci/circleci: pytorch_linux_trusty_pynightly_test
4. ci/circleci: pytorch_linux_trusty_py3_6_gcc4_8_test
5. ci/circleci: pytorch_linux_xenial_cuda8_cudnn7_py3_NO_AVX2_test
6. ci/circleci: pytorch_linux_xenial_cuda8_cudnn7_py3_test
7. ci/circleci: pytorch_linux_xenial_cuda8_cudnn7_py3_NO_AVX_NO_AVX2_test
8. ci/circleci: pytorch_linux_xenial_cuda9_2_cudnn7_py3_gcc7_test
9. ci/circleci: pytorch_linux_xenial_cuda9_cudnn7_py2_test
10. ci/circleci: pytorch_linux_xenial_py3_clang5_asan_test
11. ci/circleci: pytorch_linux_trusty_py3_6_gcc5_4_test
12. pr/caffe2-py2-devtoolset7-rocmrpm-centos7.5-test
13. pr/caffe2-py2-clang7-rocmdeb-ubuntu16.04-test
14. ci/circleci: caffe2_py2_mkl_ubuntu16_04_test
15. ci/circleci: caffe2_py2_gcc4_8_ubuntu14_04_test
16. ci/circleci: caffe2_onnx_py2_gcc5_ubuntu16_04_test
17. ci/circleci: caffe2_py2_cuda8_0_cudnn7_ubuntu16_04_test
18. ci/circleci: caffe2_py2_cuda9_1_cudnn7_ubuntu16_04_test
19. ci/circleci: caffe2_py2_cuda9_0_cudnn7_ubuntu16_04_test
20. ci/circleci: caffe2_py2_cuda9_0_cudnn7_centos7_test
21. ci/circleci: pytorch_linux_xenial_cuda8_cudnn7_py3_multigpu_test
22. ci/circleci: caffe2_cmake_cuda9_0_cudnn7_ubuntu16_04_test
23. ci/circleci: binary_linux_manywheel_2.7mu_cpu_test
```

Analysis:

- 0-11: PyTorch tests
    - 0-4: CPU

    - 5-9: CUDA
    - 10: ASAN
    - 11: GCC 5.4 (this is because it's the DEBUG=1 build!!)
- 12-13: ROCm
- 14-15: Caffe2 CPU
- 16: ONNX
- 17-20: Caffe2 CUDA
- 21: PyTorch Multi GPU
- 22: Caffe2 cmake

This correlation chart makes the CPU versus CUDA distinction more clear. The moral of the story is, perhaps, that you don't really need CUDA-specific builds to find build failures... but you do need them to find CUDA test failures.

Recommendation 1: Reduced set of builds/tests for PRs

Based on the data above, I recommend only running the following set of build/tests on PRs:

- PyTorch
    - CPU: `pytorch_linux_trusty_py2_7_9`
        - Selected oldest Python 2 version to ensure Python 2 coverage
    - CUDA: `pytorch-linux-xenial-cuda9-cudnn7-py3` (include `multigpu`)
    - ASAN: `pytorch_linux_xenial_py3_clang5_asan`
    - DEBUG: `pytorch_linux_trusty_py3_6_gcc5_4`
- ROCm
    - Caffe2: `pr/caffe2-py2-devtoolset7-rocmrpm-centos7.5`
        - Selected CentOS over Ubuntu, as it more accurately reflects fbcode
    - PyTorch: `pr/py2-clang7-rocmdeb-ubuntu16.04`
- Caffe2
    - CPU: `caffe2_py2_mkl_ubuntu16_04`
    - CUDA: `caffe2_py2_cuda9_1_cudnn7_ubuntu16_04`
    - ONNX: `caffe2_onnx_py2_gcc5_ubuntu16_04`
    - Clang: `caffe2_py2_clang7_ubuntu16_04`
    - CMake: `caffe2_cmake_cuda9_0_cudnn7_ubuntu16_04`
- Binaries
    - `binary_linux_manywheel_2.7mu_cpu`
- Unusual platforms
    - Caffe2 Android: `caffe2_py2_android_ubuntu16_04`
    - Caffe2 OSX: `caffe2_py2_system_macos10_13`
    - PyTorch OSX: `pytorch_macos_10_13_cuda9_2_cudnn7_py3`
    - PyTorch Windows: `pr/pytorch-win-ws2016-cuda9-cudnn7-py3`
- Other checks
    - `pytorch_short_perf_test_gpu`
    - `pytorch_doc_push`

    - Travis

Based on the same methodology as slow/xla tests, we can let users manually request a full test run using a special message in their commit message. We can also automatically request extra tests automatically based on files modified.

Evaluation methodology: See how often full configurations on master are broken this way, and reevaluate the default set of tests.

Necessary pieces

- Must be possible to run the full set if explicitly requested
    - [ci all]
- Must be possible to run specific jobs, if explicitly requested
    - [ci asdfasdfasdfasdf]
- ~~Automatically detect if I should run a job and run it even if it is not part of PR in that situation~~
    - ~~Approach 1: Replicate MODULES in PyTorch main repository, classify PRs on the fly~~
        - ~~Can invert ghninja to load MODULES from main repository~~
    - ~~Approach 2: Somehow read out labels (this is subject to race condition, also requires blah)~~
    - Not going to do this, because CircleCI does not expose branch base information

Appendix: Build status correlation methodology

Dataset: @Will Feng's ossci-job-status S3 bucket. Downloaded with:

```
aws s3 sync s3://ossci-job-status .
```

Preprocessing:

```
import json
import os
import sys
import csv
import itertools

table = []

def strip_end(text, suffix):
    if not text.endswith(suffix):
        return text
    return text[:len(text)-len(suffix)]

def process(fn):
    global keys
    global latest_time
    time = os.path.getctime(fn)
    with open(fn, 'r') as f:
        j = json.load(f)
        entry = {}
        new_keys = set()
        for k, v in j.items():
            k = strip_end(k, '_build')
            k = strip_end(k, '_test')
            k = strip_end(k, '-build')
            k = strip_end(k, '-test')
            k = strip_end(k, '_multigpu')
            k = strip_end(k, '_NO_AVX_NO_AVX2')
            k = strip_end(k, '_NO_AVX2')
            new_keys.add(k)
            if k not in entry:
                entry[k] = 1
            entry[k] *= 1 if v['status'] == 'success' else 0
        if latest_time is None or time > latest_time:
            latest_time = time
            keys = new_keys
        table.append(entry)

latest_time = None
keys = None

for fn in os.listdir('pr'):
    fn = os.path.join('pr', fn)
    process(fn)

for fn in os.listdir('master'):
    fn = os.path.join('master', fn)
    process(fn)

for i, v in enumerate(table):
    table[i] = {k: v[k] for k in set(v) & keys}

with open('combined.csv', 'w') as f:
    dw = csv.DictWriter(f, keys)
    dw.writeheader()
    dw.writerows(table)
```

Visualization:

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as spc
from matplotlib import cm as cm

# Read file into a Pandas dataframe
from pandas import DataFrame, read_csv
df = read_csv('combined.csv')

corr = df.corr().values
pdist = spc.distance.pdist(corr)
linkage = spc.linkage(pdist, method='complete')
idx = spc.fcluster(linkage, 0.1 * pdist.max(), 'distance')

columns = [df.columns.tolist()[i] for i in list((np.argsort(idx)))]
print("\n".join("{}. {}".format(i, c) for i, c in enumerate(columns)))
df = df.reindex_axis(columns, axis=1)
```

```
fig = plt.figure()
ax1 = fig.add_subplot(111)
cmap = cm.get_cmap('jet', 30)
cax = ax1.imshow(df.corr(), interpolation="nearest", cmap=cmap)
ax1.grid(True)
plt.title('Build failure Correlation')
# Add colorbar, make sure to specify tick locations to match desired ticklabels
fig.colorbar(cax, ticks=[.75,.8,.85,.90,.95,1])
plt.show()
```

Appendix: Miscellaneous

Things I noticed while reviewing scripts.

- We should track sccache stats in monitoring
- Why does multigpu and test scripts replicate the nccl/ssh nonsense (shouldn't they already be installed?)

Ideas for future work:

- Maybe we should also run correlation of build/test failure with files modified. (There are a lot of files, might be hard to get signal.)