Installation Caffe on Argon Cluster

Zhe Li (zhe-li-1@uiowa.edu)
Zhuoning Yuan (zhuoning-yuan@uiowa.edu)
Machine Learning and Optimization Research Group
Department of Computer science
The University of Iowa

1. Introduction

Caffe is a popular deep learning framework, developed by Yangqing Jia during his PhD at UC Berkeley [1]. Caffe is written by C++ and mainly designed for performing image classification (commonly use Convolutional Neural Networks, known as CNN). Caffe supports Python (called Pycaffe), matlab (called matcaffe) interfaces. Caffe can be also built in various environments, e.g. linux, windows. However, the installation guide of caffe is not very friendly to beginner. Therefore, we make this tutorial to simplify the installation process.

2. Installation

To run Caffe on cluster, it requires several prerequisites. We summarize the installation process as the following three steps:

- preparations of modules
- preparations of dependencies
- installation of Caffe

For each step, we need create an independent folder. This is crucial because the issues may cost you extra days to debug if you put everything in one folder.

Preparation of Modules

Before we install Caffe on cluster or server, we need to use some libraries in module form. In your home directory, we firstly create a folder modulefiles and create a file common inside using the command line:

```
$mkdir modulefiles
$cd modulefiles
$vim common
```

Then, we add following content into common (to tell the modules what to load):

```
module load python/2.7.13_parallel_studio-2017.1
module load cuda/8.0.44
module load matlab/R2016b
module load boost/1.63.0
module load OpenBLAS/0.2.19_gcc-5.4.0
module load hdf5/1.8.18
#module unload gcc
#module load mkl/11.2.0
```

Next, we locate .bashrc under your home directory and add the following commands into the file. This is to tell system to execute common file (when to install those modules):

```
$vim .bashrc

#add the following to .bashrc
module use -a /Users/[YourAccountName]/modulefiles
module load common
```

Note that you need to change [YourAccountName] in "/Users/[YourAccountName]/modulefiles" to your account name. At this moment, we are done with module part.

Preparation of Dependencies

There are several dependencies needed to be installed before installing Caffe:

- gflags
- glog
- leveldb
- mdb
- opencv
- snappy
- protobuf

Let's get started!!
We first check the version of gcc in our environment:

```
$gcc --version
```

If you see the result below, you are on the good track!

```
gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-11)
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

If not, unload gcc (also make sure all future installation uses gcc 4.8.5):

```
$module unload gcc
```

Installing those dependencies will essentially create some dynamic and static libraries as well as the header files. Therefore, we create several folders to save those files. In your home directory, we perform the following commands:

```
$mkdir .usr
$cd .usr
$mkdir local
$cd local
$mkdir bin include lib man share src
```

Now, let's start installing dependencies!

Firstly, go back to your home directory and create a folder named caffe-dependency where you put all dependencies here:

```
$mkdir caffe-dependency
```

1. mdb

```
$git clone https://gitorious.org/mdb/mdb.git
$cd mdb/libraries/liblmdb
```

```
#change the prefix path for installing mdb in Makefile
#as you could see, the prefix is my path to install mdb library
$vim Makefile
prefix  = /Users/[YourAccountName]/.usr/local
```

```
$make
$make install
```

2. Snappy

```
$git clone https://github.com/google/snappy.git
$cd snappy
$bash autogen.sh
$./configure --prefix=/Users/[YourAccountName]/.usr/local/
```

```
#In line 574, modify the dist_doc_DATA variable in Makefile line 574
$vim Makefile

From
“dist_doc_DATA = ChangeLog COPYING INSTALL NEWS README format_description.txt framing_format.txt”
To
“dist_doc_DATA = ChangeLog COPYING INSTALL NEWS format_description.txt framing_format.txt”
```

```
$make
$make install
```

3. OpenCV

```
$git clone https://github.com/Itseez/opencv.git
$mkdir build
$cd build
$cmake -DCMAKE_INSTALL_PREFIX=/Users/[YourAccountName]/.usr/local/ ..
$make
$make install
```

This may take 20 mins to finish!

4. Leveldb

```
$git clone https://github.com/google/leveldb.git
$make
$mv out-shared/libleveldb.* /Users/[YourAccountName]/.usr/local/lib/
$mv out-static/libleveldb.* /Users/[YourAccountName]/.usr/local/lib/
```

5. Gflag

```
$git clone https://github.com/schuhschuh/gflags.git
$cd gflags
$mkdir build
$cd build
$cmake -DCMAKE_INSTALL_PREFIX=/Users/[YourAccountName]/.usr/local/ ..
```

```
#Change CMAKE_CXX_FLAGS in CMakeCache.txt
$vim CMakeCache.txt
...
CMAKE_CXX_FLAGS:STRING=-fPIC
BUILD_SHARED_LIBS:BOOL = ON
BUILD_STATIC_LIBS:BOOL = ON
…
```

```
$make
$make install
```

6. Glog

```
$git clone https://github.com/google/glog.git
$cd glog
$mkdir build
$cd build
```

```
#$mkdir build, cd build, cmake -DCMAKE_INSTALL_PREFIX=/Users/zli79/.usr/local/ ..
$module load cmake/3.7.2
$cmake -DCMAKE_INSTALL_PREFIX=/Users/[YourAccountName]/.usr/local/
-DBUILD_SHARED_LIBS=ON ..
```

```
#build shared libraries
$make
$make install
```

7. Protobuf

```
$wget https://github.com/google/protobuf/releases/download/v2.5.0/protobuf-2.5.0.tar.gz
$tar -xzvf protobuf-2.5.0.tar.gz
$cd protobuf-2.5.0
$bash autogen.sh
$./configure --prefix=/Users/[YourAccountName]/.usr/local/
$make
$make install
```

Just one more thing!

So far, we have finished all dependencies installation. Before we move to next step, we need to setup proper environment variables. For convenience and simplicity, we will put the proper environment variables in a folder. The instructions are shown as below:

```
$cd modulefiles
$mkdir depends_[YourAccountShortName]
$cd depends_[YourAccountShortName]
```

```
$vim 1.0
#add the following to the 1.0 file
 set root /Users/[YourAccountName]/.usr/local/
 prepend-path    PATH                $root/bin
 prepend-path    CPLUS_INCLUDE_PATH      $root/include
```

```
prepend-path    C_INCLUDE_PATH          $root/include
prepend-path    LD_LIBRARY_PATH         $root/lib
prepend-path    LIBRARY_PATH            $root/lib
prepend-path    MANPATH                 $root/share
```

We do the same thing for .bash_profile under home directory:

```
$vim .bash_profile
#add the following to the .bash_profile
module load depends_[YourAccountShortName]/1.0
```

Note that after you configured .bash_profile, re-login your account to Argon!

Installing Caffe on Argon Cluster

```
$git clone https://github.com/BVLC/caffe.git
$cd caffe
$cp Makefile.config.example Makefile.config
```

```
#In Makefile.config, uncomment and modify these lines
$vim Makefile.config

…

OPENCV_VERSION := 3

CUDA_DIR := /opt/apps/cuda/8.0.44

#-gencode arch=compute_20,code=sm_20 \
#-gencode arch=compute_20,code=sm_21 \

BLAS := open

BLAS_INCLUDE := /opt/apps/OpenBLAS/0.2.19_gcc-5.4.0/include
BLAS_LIB := /opt/apps/OpenBLAS/0.2.19_gcc-5.4.0/lib

MATLAB_DIR := /opt/apps/matlab/R2016b

PYTHON_INCLUDE := /usr/include/python2.7 \
/opt/apps/python/2.7.13_parallel_studio-2017.1/lib/python2.7/site-packages/numpy-1.11.2-py2.7
-linux-x86_64.egg/numpy/core/include

PYTHON_LIB := /usr/lib

INCLUDE_DIRS := $(PYTHON_INCLUDE) /Users/[YourAccountName]/.usr/local/include
```

```
LIBRARY_DIRS := $(PYTHON_LIB) /Users/[YourAccountName]/.usr/local/lib
/opt/apps/boost/1.63.0/lib/ /opt/apps/hdf5/1.8.18/lib/ /usr/lib64

…
```

Let's check to make sure all dependencies are installed correctly!

1) all *.h header files construct successfully under /.usr/local/include/
2) all *.a header files construct successfully under /.usr/local/lib/

Now, let's compile Caffe:

```
$make all
$make test
$make runtest
$make distribute
$make pycaffe
```

Note that even though it comes with several errors when we run make runtest, we can still run Caffe.

3. Error Diagnostics

Congratulations! If you successfully install Caffe by following our tutorial, you are ready to go now. If not, carefully read this section and it may help resolve your problems.

First Round of Installation of Module and Dependencies

1. Check location of cuda

```
%Before load cuda module
$which cuda
/usr/bin/which: no cuda in
(/opt/sge/bin:/opt/sge/bin/lx-amd64:/opt/sge/bin:/opt/sge/bin/lx-amd64:/usr/lib64/qt-3.3/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/Users/zli79/.local/bin:/Users/zli79/bin)
```

```
%after load cuda module
$module load cuda
$which cuda
/opt/apps/cuda/8.0.44/bin:/opt/apps/cuda/8.0.44/samples/bin/x86_64/linux/release:/opt/sge/bin:/opt/sge/bin/lx-amd64:/opt/sge/bin:/opt/sge/bin/lx-amd64:/usr/lib64/qt-3.3/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/Users/zli79/.local/bin:/Users/zli79/bin
```

In the second case, we can see that cuda in **/opt/apps/ subfolder.**

2. Check gcc version in Cluster

```
$gcc --version
gcc (GCC) 5.4.0
$ uname -a
Linux argon-login-2.hpc 3.10.0-514.2.2.el7.x86_64 #1 SMP Tue Dec 6 23:06:41 UTC 2016
x86_64 x86_64 x86_64 GNU/Linux
```

3. Install Snappy

```
$bash autogen.sh
$./configure --prefix=/Users/zli79/.usr/local/
$make
$Make install
```

Error:

```
/usr/bin/install: cannot stat ‘./README’: No such file or directory
make[1]: *** [install-dist_docDATA] Error 1
make[1]: Leaving directory `/Users/zli79/caffe-dependency-package/snappy'
make: *** [install-am] Error 2
```

Workaround:

This is not a solution, I just delete README from `dist_doc_DATA` variable

From

```
dist_doc_DATA = ChangeLog COPYING INSTALL NEWS README format_description.txt
framing_format.txt
```

To

```
dist_doc_DATA = ChangeLog COPYING INSTALL NEWS format_description.txt
framing_format.txt
```

4. Install protobuf (2.5.0 version)

```
$wget https://github.com/google/protobuf/releases/download/v2.5.0/protobuf-2.5.0.tar.gz
$bash autogen.sh
$./configure --prefix=/Users/zli79/.usr/local/
$make
$make install
```

It seems working!

5. Install OpenCV

```
$mkdir build
$cd build
```

```
$cmake -DCMAKE_INSTALL_PREFIX=/Users/zli79/.usr/local/ ..
$make
$make install
```

6. Install Leveldb

```
$make
```

Not working for the following commands!

```
$cp --preserve=links out-shared/libleveldb.* /Users/zli79/.usr/local/lib/
$cp --preserve=links out-static/libleveldb.* /Users/zli79/.usr/local/lib/
$cp -r include/leveldb/* /Users/zli79/.usr/local/include/
```

Alternative solution works!

```
$mv out-shared/libleveldb.* /Users/zli79/.usr/local/lib/
$mv out-static/libleveldb.* /Users/zli79/.usr/local/lib/
```

7.glog

```
$./configue --prefix=/Users/zli79/.usr/local/
$make
```

Error:

```
CDPATH="${ZSH_VERSION+.}:" && cd . && aclocal-1.14 -I m4
/bin/sh: aclocal-1.14: command not found
make: *** [aclocal.m4] Error 127
[zli79@argon-login-1 glog]$ grep -wrn ./ -e "aclocal-1.14"
./Makefile:613:ACLOCAL = aclocal-1.14
./config.log:2068:ACLOCAL='aclocal-1.14'
./config.status:995:S["ACLOCAL"]="aclocal-1.14"
```

Attempt:

Modify `configure :`

From

```
am__api_version='1.14'
```

To

```
am__api_version='1.13.4'
```

Error:

```
$make
$cd . && automake-1.13.4 --gnu
/bin/sh: line 4: automake-1.13.4: command not found
make: *** [Makefile.in] Error 1
```

Attempt:

```
$autoreconf --force --install
$make
```

Error:

```
/opt/apps/binutils/2.27/bin/ld: /Users/zli79/.usr/local/lib/libgflags.a(gflags_completions.cc.o):
relocation R_X86_64_32S against symbol
`_ZNSs4_Rep20_S_empty_rep_storageE@@GLIBCXX_3.4' can not be used when making a
shared object; recompile with -fPIC
/opt/apps/binutils/2.27/bin/ld: final link failed: Nonrepresentable section on output
collect2: error: ld returned 1 exit status
make: *** [libglog.la] Error 1
```

My thought is that since to create glog dynamic or static library, we need to "ld" gflag, remember when compiling gflag, we use the flag -fPIC.

Error:

```
/Users/zli79/caffe-dependency-package/glog/src/googletest.h:94: undefined reference to
`google::FlagRegisterer::FlagRegisterer<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > >(char const*, char const*, char const*,
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >*,
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >*)'
collect2: error: ld returned 1 exit status
make: *** [logging_unittest] Error 1
```

When I use the following command:

```
$nm -C libgflags.a | grep FlagRegisterer
%Which lists the functions in libgflags.a as
0000000000000000 W google::FlagRegisterer::FlagRegisterer<bool>(char const*, char const*,
char const*, bool*, bool*)
0000000000000000 W google::FlagRegisterer::FlagRegisterer<double>(char const*, char
const*, char const*, double*, double*)
```

From this point, that is to say, the function is over there. We must link it properly.

Solution:

I have tried a lot of possible solutions on this and eventually, I use cmake to compile and install glog.

```
$mkdir build
$cd build
%$mkdir build, cd build, cmake -DCMAKE_INSTALL_PREFIX=/Users/zli79/.usr/local/ ..
$cmake -DCMAKE_INSTALL_PREFIX=/Users/zli79/.usr/local/ -DBUILD_SHARED_LIBS=ON ..
%build shared libraries
$make
```

```
$make install
```

Be careful about the Cmake version. So we do before we go further:

```
$module load cmake/3.7.2
```

8. Gflag

```
$mkdir build, cd build, cmake -DCMAKE_INSTALL_PREFIX=/nfs/01/cwr0463/.usr/local/ ..
$vim CMakeCache.txt
%Change the following
CMAKE_CXX_FLAGS:STRING=-fPIC
$make
$make install
```

8. Git clone caffe

```
$git clone https://github.com/BVLC/caffe.git
$cp Makefile.config.example Makefile.config
Then we need to configure the Makefile.config properly.
```

9. Python library compilation

```
$for req in $(cat requirements.txt); do pip install --user $req; done
```

Compiling caffe:

```
$make
```

Error 1:

```
make: protoc: Command not found
make: *** [.build_release/src/caffe/proto/caffe.pb.cc] Error 127
```

Solution:

```
%add protoc to the PATH
$export PATH=$PATH:/Users/zli79/.usr/local/bin/
```

Errors 2:

```
CXX .build_release/src/caffe/proto/caffe.pb.cc
In file included from .build_release/src/caffe/proto/caffe.pb.cc:5:0:
.build_release/src/caffe/proto/caffe.pb.h:9:42: fatal error: google/protobuf/stubs/common.h: No
such file or directory
 #include <google/protobuf/stubs/common.h>
                                          ^
compilation terminated.
```

```
make: *** [.build_release/src/caffe/proto/caffe.pb.o] Error 1
```

Solution:
Firstly, we check

```
$ls ~/.usr/local/include/google/protobuf/stubs/
```

Since common.h file is there, so the problem is that the include path is not correct! Modify the following in Makefile.config:

```
From
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include
To
INCLUDE_DIRS := $(PYTHON_INCLUDE) /Users/zli79/.usr/local/include
```

10. Make all

```
$make all
```

Error:

```
make: /usr/local/matlab/R2016b//bin/mexext: Command not found
```

Solution:
We found the file is in /opt/apps/matlab/R2016b/bin/mexext .so, we modify Makefile.configure:

```
MATLAB_DIR :=/opt/apps/matlab/R2016b
```

Warning:

```
nvcc warning : The 'compute_20', 'sm_20', and 'sm_21' architectures are deprecated, and may be removed in a future release (Use -Wno-deprecated-gpu-targets to suppress warning).
```

Solution:
Since we use cuda-8.0 version, so compute_20 and sm_20 is compatible with old version of cuda. Just remove CUDA_ARCH variable compute_20 sm_20.

Warning:

```
/opt/apps/boost/1.63.0/include/boost/smart_ptr/detail/sp_counted_base_gcc_x86.hpp(75): warning: variable "tmp" was set but never used
```

Solution:
Have not found solution, so far it is warning!

Error:

```
.build_release/lib/libcaffe.so: undefined reference to `leveldb::Status::ToString[abi:cxx11]() const'
.build_release/lib/libcaffe.so: undefined reference to `google::protobuf::Message::DebugString[abi:cxx11]() const'
.build_release/lib/libcaffe.so: undefined reference to `google::protobuf::MessageFactory::InternalRegisterGeneratedFile(char const*, void (*)(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&))'
```

Solution:

It seems that protobuf is not compiled correctly. And reference is here: http://stackoverflow.com/questions/30124264/undefined-reference-to-googleprotobufinternalempty-string-abicxx11

Error:

```
CXX/LD -o .build_release/tools/finetune_net.bin
.build_release/lib/libcaffe.so: undefined reference to `google::base::CheckOpMessageBuilder::NewString[abi:cxx11]()'
collect2: error: ld returned 1 exit status
make: *** [.build_release/tools/finetune_net.bin] Error 1
```

Solution:

Two things left I could try are
1) change `gcc` version
2) copy `.so` files from dual server.

Error:

```
/Users/zli79/.usr/local/lib/libopencv_imgcodecs.so.3.2: error adding symbols: DSO missing from command line
collect2: error: ld returned 1 exit status
make: *** [.build_release/examples/cpp_classification/classification.bin] Error 1
```

Solution:

It could be that you are using OpenCV version 3. If yes, just uncomment the following line in your `Makefile.config`:

From

```
# OPENCV_VERSION := 3
```

To

```
OPENCV_VERSION := 3
```

So far, it seems that “make all” works well!

12. Make test

```
$make test
```

Error:

```
CXX/LD -o .build_release/test/test_all.testbin src/caffe/test/test_caffe_main.cpp
.build_release/src/caffe/test/test_benchmark.o: In function `void
boost::this_thread::sleep<boost::date_time::subsecond_duration<boost::posix_time::time_durati
on, 1000l> >(boost::date_time::subsecond_duration<boost::posix_time::time_duration, 1000l>
const&)':
test_benchmark.cpp:(.text._ZN5boost11this_thread5sleepINS_9date_time18subsecond_duratio
nINS_10posix_time13time_durationELl1000EEEEEvRKT_[_ZN5boost11this_thread5sleepINS_
9date_time18subsecond_durationINS_10posix_time13time_durationELl1000EEEEEvRKT_]+0x
27d): undefined reference to `boost::this_thread::hidden::sleep_until(timespec const&)'
collect2: error: ld returned 1 exit status
make: *** [.build_release/test/test_all.testbin] Error 1
```

Solution:

Libboost_thread.so file is in the following:

```
/opt/apps/boost/1.63.0/lib/libboost_thread.*
/usr/lib64/libboost_thread-mt.so
/usr/lib64/libboost_thread.so
/usr/lib64/libboost_thread-mt.so.1.53.0
```

Right now, in the system it seems that there are two version of boost Libboost_filesystem.so.1.53.0 and libboost_system.so.1.63.0.

Warning:

```
/opt/apps/binutils/2.27/bin/ld: warning: libboost_system.so.1.53.0, needed by
.build_release/lib/libcaffe.so, may conflict with libboost_system.so.1.63.0
/opt/apps/binutils/2.27/bin/ld: warning: libboost_filesystem.so.1.53.0, needed by
.build_release/lib/libcaffe.so, may conflict with libboost_filesystem.so.1.63.0
```

Solution:

We need take care of the boost version. Find and modify Makefile in line 274:

```
From
LIBRARIES += boost_thread stdc++
To
LIBRARIES += boost_thread-mt stdc++
```

13. Make runtest

```
$make runtest
```

Error:

```
.build_release/tools/caffe
.build_release/tools/caffe: error while loading shared libraries: libhdf5_hl.so.8: cannot open shared object file: No such file or directory
make: *** [runtest] Error 127
```

Solution:

Check the hdf5 module --different version

```
Before we compile, the module load information
OpenBLAS/0.2.19_gcc-5.4.0                    (L)
boost/1.63.0                                 (D)
cuda/8.0.44                                  (L)
curl/7.52.1                                  (L,D)
hdf5/1.8.18                                  (D)
matlab/R2016b                                (L)
python/2.7.13_parallel_studio-2017.1         (L)
python/3.5.3_parallel_studio-2017.1          (D)
```

Error:

```
Could not open or find file examples/images/cat.jpg
F0322 22:32:19.510113 40173 image_data_layer.cpp:77] Check failed: cv_img.data Could not load examples/images/cat.jpg
```

Analysis:

We found the libjpeg.so file is under:

/usr/lib64/libjpeg.so
/usr/lib64/libjpeg.so.62
/usr/lib64/libjpeg.so.62.1.0

Someone says non-compatible version of libjpeg in the system. The following is from dual server:

/usr/lib64/libjpeg.so
/usr/lib64/libjpeg.so.62
/usr/lib64/libjpeg.so.62.1.0

14. Make pycaffe

```
$make pycaffe
```

Error:

```
python/caffe/_caffe.cpp:10:31: fatal error: numpy/arrayobject.h: No such file or directory
 #include <numpy/arrayobject.h>
                           ^
compilation terminated.
make: *** [python/caffe/_caffe.so] Error 1
```

Analysis:

```
$module avail
python/2.7.13_parallel_studio-2017.1                (L)
```

We could find that file by

```
$python
>>>import numpy
>>>numpy.get_include()
/opt/apps/python/2.7.13_parallel_studio-2017.1/lib/python2.7/site-packages/numpy-1.11.2-py2.7
-linux-x86_64.egg/numpy/core/include
$ls
/opt/apps/python/2.7.13_parallel_studio-2017.1/lib/python2.7/site-packages/numpy-1.11.2-py2.7
-linux-x86_64.egg/numpy/core/include/numpy/arrayobject.h
```

Solution:

Modify Makefile.config:

From

```
PYTHON_INCLUDE := /usr/include/python2.7 \
 71                /usr/lib/python2.7/dist-packages/numpy/core/include #original, make all test works
```

To

```
PYTHON_INCLUDE := /usr/include/python2.7 \
 74
/opt/apps/python/2.7.13_parallel_studio-2017.1/lib/python2.7/site-packages/numpy-1.11
.2-py2.7-linux-x86_64.egg/numpy/core/include
```

Summary, at this point, even there is problem related to libjpeg.so dynamic library problem. But when I try to train caffe using P100 GPU, it works!

Second Round of Compiling Caffe and Installation

Error:

```
libopencv_imgcodecs.so: undefined reference to `jpeg_finish_decompress@LIBJPEG_6.2'
```

```
collect2: error: ld returned 1 exit status
make: *** [.build_release/tools/upgrade_net_proto_binary.bin] Error 1
```

Analysis:

Since there is libjpeg.so which is located ~/.usr/local/lib/

Solution:

I just explicitly delete those, then it works.

```
$rm libjpeg.so.*
```

When I submit the job to GPU-node, it sometimes prompts the following, which reduce the speed of training:

```
blocking_queue.cpp:49 Waiting for data
```

```
$iostat -dx /dev/sda

Linux 3.10.0-514.10.2.el7.x86_64 (optimus.cs.uiowa.edu)
03/24/2017          _x86_64_            (40 CPU)
Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await
w_await  svctm  %util
sda               1.09     4.40    3.96    5.62    99.17    94.64    40.43     0.05    5.10    1.32
0.13    0.12
```

In order to use cudnn optimization, we copy cuda/lib/libcudnn.so libcudnn.so.5 libcudnn.so.5.0.5 libcudnn_static.a to ~/.usr/local/lib/ and copy cuda/include/cudnn.h to ~/.usr/local/include/ on argon server. Now, make all and make test works well!

On argon cluster, we need to check the output of

```
I0324 13:22:46.746095 40121 solver.cpp:238]     Train net output #0: logprob = 1.94882 (* 1 =
1.94882 loss)
I0324 13:22:46.746104 40121 solver.cpp:238]     Train net output #1: loss1_softmax = 3.00236
(* 0.3 = 0.900708 loss)
I0324 13:22:46.746110 40121 solver.cpp:238]     Train net output #2: loss2_softmax = 2.01386
(* 0.3 = 0.604159 loss)
I0324 13:22:46.746114 40121 solver.cpp:238]     Train net output #3: top1 = 0.554688
I0324 13:22:46.746117 40121 solver.cpp:238]     Train net output #4: top5 = 0.8125
```

Reference

[1] Caffe official documentation. http://caffe.berkeleyvision.org/
[2] Install Caffe On The Ohio Super Computing (OSC) Ruby Cluster.
http://www.andrewjanowczyk.com/installing-caffe-on-the-ohio-super-computing-osc-ruby-cluster/