Kaldi installation document on the new cluster

1. Download the Kaldi open source from Github

$ git clone <https://github.com/kaldi-asr/kaldi.git>

1. Entry tools directory and compile it

$ cd tools && make -j num\_jobs

For example: $ cd tools && make -j 20

1. Entry src directory and compile it

$ ./configure –use-cuda –cudatk-dir=/path-to-cuda\

--mkl-root=/path-to-mkl –mkl libdir=/path-to-mkl-liborary

$ make clean -j num\_jobs

$ make depend -j num\_jobs

$ make -j num\_jobs

For example :

$ ./configure --use-cuda --cudatk-dir=/cm/shared/apps/cuda10.0/toolkit/10.0.130 \

--mkl-root=/cm/shared/apps/mkl\_intel/mkl \

–mkl libdir=/cm/shared/apps/mkl\_intel/mkl/lib/intel64

$ make clean -j 15

$ make depend -j 25

$ make -j 30

Note: if you want to train a acoustics model with using CUDA in the Kaldi . you must be specify the argparse the above example when compile it, CUDA library and mkl library is not default installation path in the new class, so you must specify them.

1. Check if cuda is used to compile Kaldi

$ cd src/cudamatrix

$ make test -j 5

If you see the below output, it will represent that Kaldi is compiled with CUDA.

Running cu-vector-test ... 13s... SUCCESS cu-vector-test

Running cu-matrix-test ... 20s... SUCCESS cu-matrix-test

Running cu-math-test ... 17s... SUCCESS cu-math-test

Running cu-test ... 6s... SUCCESS cu-test

Running cu-sp-matrix-test ... 6s... SUCCESS cu-sp-matrix-test

Running cu-packed-matrix-test ... 5s... SUCCESS cu-packed-matrix-test

Running cu-tp-matrix-test ... 6s... SUCCESS cu-tp-matrix-test

Running cu-block-matrix-test ... 9s... SUCCESS cu-block-matrix-test

Running cu-matrix-speed-test ... 61s... SUCCESS cu-matrix-speed-test

Running cu-vector-speed-test ... 17s... SUCCESS cu-vector-speed-test

Running cu-sp-matrix-speed-test ... 4s... SUCCESS cu-sp-matrix-speed-test

Running cu-array-test ... 3s... SUCCESS cu-array-test

Running cu-sparse-matrix-test ... 3s... SUCCESS cu-sparse-matrix-test

Running cu-device-test ... 7s... SUCCESS cu-device-test

Running cu-rand-speed-test ... 4s... SUCCESS cu-rand-speed-test

Running cu-compressed-matrix-test ... 3s... SUCCESS cu-compressed-matrix-test

1. (Option), if you want to ngram-count command to get n-gram language model, you need run the tools (kald/tools/ install\_srilm.sh). sirlm is not a completely open source tool, Kaldi will not automatically install and compile by default, so you need to download it manually from its official website(link is <http://www.speech.sri.com/projects/srilm/download.html>), however, I have download the source code to the new class(/home4/md510/package/srilm.tgz), you copy the srilm.tgz to your kaldi/tools, then you running the command and compile the language model:

$ ./ install\_srilm.sh

GCC installation document on the new class without sudo

1. Download the open source from the website(https://ftp.gnu.org/gnu/gcc/)

For example: gcc7.5.0

$ wget <https://ftp.gnu.org/gnu/gcc/gcc-7.5.0/gcc-7.5.0.tar.gz>

1. Unzip it

$ tar xzf gcc-7.5.0.tar.gz

1. Download various dependencies

$ cd gcc-7.5.0

$ ./contrib/download\_prerequisites

1. Compile it

$ cd ..

$ mkdir objdir

$ cd objdir

$PWD/../gcc-7.3.0/configure --prefix=$HOME/gcc-7.3.0 --enable-languages=c,c++,fortran,go

For example: /home4/md510/package/gcc-7.3.0/configure --prefix=$HOME/gcc-7.3.0 --enable-languages=c,c++,fortran,go

$ make -j 10

$ make install

1. Add the gcc version to your environment.

$ vim ~/.bashrc

Then add the below content to the ~/.bashrc and save them

export PATH=~/gcc-7.5.0/bin:$PATH

export LD\_LIBRARY\_PATH=~/gcc-7.5.0/lib:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=~/gcc-7.5.0/lib64:$LD\_LIBRARY\_PATH

$ source ~/.bashrc

1. Check if the gcc is used.

$ gcc -v