AMD

ROCm

# Chapter 4.3: Converting CUDA KMeans to HIP

**ROCm Tutorial | AMD 2020**

AMD
ROCm

# Table of Contents

AMD ROCm

# Chapter 4.3: Converting CUDA KMeans to HIP

This hands-on tutorial shows how we can convert a publicly available KMeans application which is written in CUDA to HIP.

## Preparation

1. Move to the working directory:

   ```
   cd Chapter4/03_Cuda_To_HIP_Kmeans
   ```

2. The CUDA source files can be found in CUDA_Kmeans

3. No libraries are required to install for this particular example

4. We can move straight to use the HIP tools for the source conversion

5. Copy the source files to the HIP_KMeans directory:

   ```
   cp -r CUDA_KMeans/* HIP_Kmeans
   ```

6. Copy the provided Makefile Makefile.hip which is located inside 03_Cuda_To_HIP_KMeans to HIP_Kmeans

   ```
   cp Makefile.hip HIP_Kmeans
   ```

## Scanning and Porting

Similar to the previous two examples, run the scripts “hipexamine-perl.sh” and “hipconvertinplace-perl.sh” to scan and port the files from CUDA to HIP using the inbuilt conversion tools

AMD ROCm

## Makefile Change

1. Similar to CNN example, we will need to now manually change the Makefile to support building for HIP

2. For convenience you can use the provided Makefile.hip to compile

3. If you compare Makefile and Makefile.hip you will observe, the following changes:
    - nvcc invocations to hipcc
    - Any nvcc or CUDA specific flags have been replaced with ROCm specific flags

## Compiling

1. Run make -f Makefile.hip
2. This will build your application
3. However, you will encounter the error shown in Figure 1
4. This error seems to suggest in kmeans.h file, the HIP header is missing
5. Include “hip/hip_runtime.h” to the kmeans.h file
6. Now again run make -f Makefile.hip. The application will compile successfully

```
1 warning and 3 errors generated.
In file included from cuda_main.cpp:51:
./kmeans.h:53:23: error: unknown type name 'hipError_t'
inline void checkCuda(hipError_t e) {
                      ^
./kmeans.h:54:14: error: use of undeclared identifier 'hipSuccess'
    if (e != hipSuccess) {
             ^
./kmeans.h:63:15: error: use of undeclared identifier 'hipGetLastError'
    checkCuda(hipGetLastError());
              ^
cuda_main.cpp:56:9: warning: ISO C++11 does not allow conversion from string literal to
        "Usage: %s [switches] -i filename -n num_clusters\n"
        ^
1 warning and 3 errors generated.
In file included from cuda_io.cpp:26:
./kmeans.h:53:23: error: unknown type name 'hipError_t'
inline void checkCuda(hipError_t e) {
                      ^
./kmeans.h:54:14: error: use of undeclared identifier 'hipSuccess'
    if (e != hipSuccess) {
             ^
./kmeans.h:63:15: error: use of undeclared identifier 'hipGetLastError'
    checkCuda(hipGetLastError());
              ^
3 errors generated.
In file included from cuda_io.cpp:26:
./kmeans.h:53:23: error: unknown type name 'hipError_t'
inline void checkCuda(hipError_t e) {
                      ^
./kmeans.h:54:14: error: use of undeclared identifier 'hipSuccess'
    if (e != hipSuccess) {
             ^
./kmeans.h:63:15: error: use of undeclared identifier 'hipGetLastError'
    checkCuda(hipGetLastError());
              ^
```

**Figure 1: Compilation Error**

# Executing

1. Compiling will generate the binary : ./hip_main

   ./hip_main will show the possible options for this application

2. Let's run an example:

   ./hip_main -i Image_data/edge100.txt -n 4

3. Again, with minimal effort we were able to port a full CUDA application to HIP