ECE 786 Programming Assignment 1 PartB Specification

Description:

In the docker container, we have installed the necessary cuda toolkits and libs that used to build the simulator. We use gpgpusim-4.0.1 built with cuda toolkit-10.1 for our course assignment.

Build the simulator using Docker

1. Steps of creating your first docker container

- Use Docker to run a container where GPGPU-Sim is already installed. Please follow the instructions on the official Docker Website and install Docker on your machine accordingly. If the version of your operating system is not high enough to install Docker, please install “Docker Toolbox” instead.
- After Docker has been fully installed on your machine, open a terminal window. For those of you who have installed “Docker Toolbox" instead, open a “Docker Quick Start Terminal”.
- Run the following command to pull the docker image where GPGPU-Sim is already installed:
  docker pull pli11/gpgpusim:cuda_10_1
- Run the following command to start the container
  docker run -w /root -it pli11/gpgpusim:cuda_10_1 /bin/bash
- Run “ls”, and you will see there is one directory “gpgpu-sim_distribution”. That is where the GPGPU-Sim source code resides.
- Go to the directory “gpgpu-sim_distribution”, checkout tag v4.0.1 and create a branch v401 for use.
  cd ~/gpgpu-sim_distribution/
  git checkout tags/v4.0.1 -b v401
- Run the following command to compile GPGPU-Sim:
  cd ~/gpgpu-sim_distribution/
  make clean
  source setup_environment
  make
- When you would like to exit your current container, type "exit".

2. Steps of returning to the docker container you have worked on

- As you may have noticed, every time you run the command “docker run -w /root -it pli11/gpgpusim:cuda_10_1 /bin/bash”, you will create a new container.
- To find the container (virtual machine) that you have worked on, run "docker container ls -a" and see a list of all your containers and their containerIDs.
- If you would like to enter a container, run the following command:
  docker exec -it ContainerID /bin/bash
- Make sure the container is up before you try to enter it (by running “**docker start ContainerID**”)

Compile your program and run your program with GPGPU-SIM

1) After you start the container, set up the environment.
   source ~/gpgpu-sim_distribution/setup_environment
2) Download the vectorAdd example from github.
   git clone https://github.com/peiyi1/vectorAdd.git
3) Go to the directory vectorAdd, and compile the program with nvcc flag “--cudart=shared” (You should notice the nvcc flag “--cudart=shared” was used in the Makefile)
   cd ~/vectorAdd/
   make clean
   make
4) Enter the directory ~/gpgpu-sim_distribution/configs/tested-config/SM7_QV100/ (using command “cd ~/gpgpu-sim_distribution/configs/tested-cfgs/SM7_QV100/”), and read the gpgpusim.config file carefully, try to understand the config file(since we have not discussed the gpu architecture topics a lot currently, try to understand the config options as much as you can).
5) copy the file of gpgpusim.config and config_volta_islip.icnt into the directory of vectorAdd.
   cd ~/vectorAdd/
   cp ~/gpgpu-sim_distribution/configs/tested-cfgs/SM7_QV100/* .
6) Run the program to get the simulation output of vectorAdd.
   ./vectorAdd > output.txt
7) Replace the file of vectorAdd and use your own CUDA code in Program Assignment PartA. Re-compile your CUDA code(the cudaMalloc version) with nvcc flag “--cudart=shared”. Do not run your cudaMallocManaged() code on the simulator cause the unified memory management is not supported by the simulator now.
8) Run your program with the functional simulator (modify the file of gpgpusim.config to change the option -gpgpu_ptx_sim_mode to 1), and verify the correctness of your output with the sampleInput.

9) Run your program with the performance simulator (modify the file of gpgpusim.config to change the option -gpgpu_ptx_sim_mode to 0), and try to analysis the following statistics with the sampleInput:
   i. What is the IPC of your program and how is this value calculated from the statistics?
   ii. What is the data cache miss_rate and how is this value calculated from the statistics?
   iii. You should be able to find these statistics directly from the simulator output.

Appendix

Below are some resources for learning GPGPU-Sim. Although they are written based on the old version of GPGPU-Sim, they are still good references to learn.

- gpgpu-sim manual:
  http://gpgpu-sim.org/manual/index.php/Main_Page
- gpgpu-sim code study:
  http://people.cs.pitt.edu/~yongli/notes/gpgpu/GPGPUSIMNotes.html