CUDA Tutorial 01 - Ocelot

Installation and usage of the GPU emulator Ocelot with the CUDA toolkit 5.0 under Linux

Martin A. Kruse - June 2013

The full series of tutorials can be found online at
http://www.ieap.uni-kiel.de/et/people/kruse/,
though access to this site later than 2026 might
be impossible due to thermonuclear war.

Suggestions and corrections are very welcome at kruse@physik.uni-kiel.de.

GPU Ocelot is a powerful tool for emulating CUDA enabled devices. It allows you to program and debug CUDA programs without the need for a CUDA device installed in your computer. This tutorial is meant to get you up and running with the CUDA computing platform utilizing Linux and the GPU emulation software GPU Ocelot. This tutorial has been tested with Ubuntu 11.10 desktop 64 bit. Although the cuda runtime version installed during this tutorial is 5.0 (up to compute capability 3.5), the emulator software so far only supports runtime version 4.0 (up to compute capability 2.1). If you have no idea what that means, just ignore it.

1 Installation

The installation procedure is quite painstaiking. Although there are packages available online at the project homepage (http://code.google.com/p/gpuocelot/) for Ubuntu 11.10, a simple installation was not possible for me. Instead a custom build was necessary. Please consult the project homepage and find out, if your linux distribution is supported and how to install gpuocelot. Maybe the installation packages work for you. Just try them.

The procedure described below is what worked for me on a clean Ubuntu installation, though as there are many package installations involved, more current software might render this procedure useless. This tutorial is based upon the installation manual found at the GPU Ocelot project page (http://code.google.com/p/gpuocelot/downloads/detail?name=Ocelot_Installation_Manual_August_2012.pdf) and the tutorial found at http://barefeg.wordpress.com/2012/06/16/how-to-install-gpuocelot-in-ubuntu-12-04/. Good luck to you!

1.1 Required tools

First, some dependencies have to be fulfilled. Open a terminal and type

```
sudo apt-get install flex bison scons build-essential subversion libboost-dev libboost-system-dev
    libboost-filesystem-dev libboost-thread-dev libglew1.6-dev
sudo apt-get install freeglut3 freeglut3-dev
```

The second command is necessary for OpenGL-samples provided with gpuocelot and is therefore optional.

1.2 Checkout and install llvm

Although described as an optional step in the installation manual, I found that not installing llvm makes the installation process of GPU Ocelot abort. In your terminal, navigate to some arbitrary location of your liking (e.g. your desktop) and type the following commands. It will take some time. The revision I checked out is 183394.

```
svn co http://llvm.org/svn/llvm-project/llvm/trunk llvm
cd ./llvm
sudo ./configure
sudo make install
```

1.3 Checkout and install GPU Ocelot

You can choose between only installing the gpuocelot program or grabbing a package of samples in the process. The second option consumes quite a lot of disk space (1.5GB), however, the samples may help you in understanding the paradigm with which gpuocelot is utilized. I checked out revision 2235.

In your terminal, navigate to some arbitrary location and type for the full package with samples

```
svn checkout http://gpuocelot.googlecode.com/svn/trunk/ gpuocelot
cd ./gpuocelot/ocelot
sudo ./build.py --install
```

or for the GPU Ocelot emulator only

```
svn checkout http://gpuocelot.googlecode.com/svn/trunk/ocelot gpuocelot
cd ./gpuocelot
sudo ./build.py --install
```

1.4 Install CUDA Toolkit 5.0

Download and install from https://developer.nvidia.com/cuda-downloads. Be sure to not install the driver as this will abort the installation process if you do not have a CUDA capable device in your computer.

2 "Hello World!" program

After this installation process, a simple example program will verify that your system is ready to emulate a CUDA device. The specifics of the program given below are not explained. This is the topic of another tutorial.

2.1 Create host program

Create a new folder called "tutorial01o" and in it a file named "main.cpp". Open it with an editor and copy and paste the following text:

```
#include <stdio.h>

extern void cuda_doStuff(void);

int main( int argc, const char* argv[] )
{
    printf("Hello␣from␣main␣function...\n");
    cuda_doStuff();
}
```

2.2 Create CUDA program

Create another file named "cuda_wrapper.cu" and copy and paste the following program:

```
#include <cuda.h>
#include <cuda_runtime.h>
#include <stdio.h>

__global__ void someKernel(int N)
{
    int idx = blockIdx.x*blockDim.x + threadIdx.x;

    if (idx<N)
        printf("Hello␣from␣thread␣#␣%i␣(block␣#:␣%i)\n", idx, blockIdx.x);
}

extern void cuda_doStuff(void)
{
    int numberOfBlocks = 2;
    int threadsPerBlock = 5;
    int maxNumberOfThreads = 10;
    printf("Hello␣from␣CUDA␣wrapper␣function...\n");
    someKernel<<<numberOfBlocks, threadsPerBlock>>>(maxNumberOfThreads);
    cudaDeviceSynchronize();
}
```

2.3 Create Ocelot configuration file

Create a file named "configure.ocelot" and paste the following text:

```
{
    ocelot: "ocelot",
    trace: {
        database: "traces/database.trace",
        memoryChecker: {
            enabled:             false,
            checkInitialization: false
        },
        raceDetector: {
            enabled:                false,
            ignoreIrrelevantWrites: false
        },
        debugger: {
            enabled:      false,
            kernelFilter: "",
            alwaysAttach: true
        }
    },
    cuda: {
        implementation: "CudaRuntime",
        tracePath:      "trace/CudaAPI.trace"
    },
    executive: {
        devices:                  [emulated],
        optimizationLevel:        full,
        defaultDeviceID:          0,
        asynchronousKernelLaunch: True,
        port:                     2011,
        host:                     "127.0.0.1",
        workerThreadLimit:        2,
        warpSize:                 32
    },
    optimizations: {
        subkernelSize:        10000,
    }
}
```

This file essentially controls the emulation of the GPU. If you need to simulate specific devices or have to debug some device code, this is the place to go. Unforunately, the documentation of Ocelot is very.... simple. Here is a very basic description of what can be done with this configuration file: http://code.google.com/p/gpuocelot/wiki/OcelotConfigFile. For preliminary testing of your program, the provided configuration file should suffice.

2.4 Create makefile

Create a file named "Makefile" and paste the following commands:

```
CC=g++
LINKER_DIRS=-L/usr/local/cuda-5.5/lib64
LINKER_FLAGS=-lcudart -lcuda
NVCC=nvcc
CUDA_ARCHITECTURE=20
OCELOT=`OcelotConfig -l`

all: main

main: main.o cuda_wrapper.o
    $(CC) main.o cuda_wrapper.o -o main $(LINKER_DIRS) $(LINKER_FLAGS) $(OCELOT)
```

```
main.o: main.cpp
    $(CC) main.cpp -c -I .

cuda_wrapper.o: cuda_wrapper.cu
    $(NVCC) -c cuda_wrapper.cu -arch=sm_$(CUDA_ARCHITECTURE)

clean:
    rm -f main.o cuda_wrapper.o main
```

3 Compilation and execution

Now everything is in place in order to emulate a CUDA device and test it with this simple program.

3.1 Compile program

Open a terminal, navigate to your project folder (tutorial01o) and type

```
make
```

3.2 Run program

In that same terminal, type

```
./main
```

The output should look something like this:

Figure 1. Output of the "Hello World" program.