CUDA Programming on NVIDIA GPUs
Mike Giles

Makefiles

Makefiles are a useful way to manage the compiling and linking of programs, especially when they are based on multiple source files. The key feature of Makefiles is that they will do the minimum amount of work necessary, so with a correctly constructed Makefile, if you have only changed one source file it will only re-compile that one file and then link everything together.

In the Makefile for the first practical, the first 3 lines define a set options which are then used in the rest of the Makefile.

The other sections each define how to "make" something. For example, the lines

```
prac1a: prac1a.cu Makefile
        nvcc prac1a.cu -o prac1a $(INC) $(NVCCFLAGS) $(LIB)
```

say that the executable file `prac1a` depends on the source file `prac1a.cu` and `Makefile`, and if either of these have changed since `prac1a` was last created then a new `prac1a` should be created using the command on the second line.

The first section

```
all: prac1a prac1b prac1c
```

is the default section and says that all three executables should be updated, if any of their dependencies have changed. It is invoked by the command

```
make
```

whereas individual executables can be generated by commands such as

```
make prac1a
```

The last section

```
clean:
        rm -f prac1a prac1b prac1c
```

which can be invoked by the command

```
make clean
```

is a convenient way to delete all of the executables.

The line

```
INC := -I$(CUDA_HOME)/include -I.
```

defines a string which tells the compiler where to look for header files, and the line

```
LIB := -L$(CUDA_HOME)/lib64 -lcudart
```

tells it where to look for libraries.

The line

```
NVCCFLAGS := -lineinfo -arch=sm_70 --ptxas-options=-v --use_fast_math
```

passes a number of "flags" to the compiler. The first generates information on line numbers which is useful is error-reporting. The second specifies that the code is to be compiled for GPUs of Compute Capability 7.0 and later. The third prints out additional information, such as the number of registers used by each kernel. The last triggers various optimisations, including the use of the SFU (Special Function Unit) for various single precision functions ($\sin, \cos, \exp, \log$).