Pre-compile the OpenCL Kernel Program - Part 2

Sat 22 November 2014
Tags opencl
Posted by Logan

In the part 1 of this article, we have mentioned how to pre-compile the OpenCL kernel program and load the pre-compiled binaries with the OpenCL API.

However, I was using the `ioc64` command from the Intel OpenCL SDK to pre-compile the kernel program. This command might be unavailable in the other OpenCL implementations. How could we get the compiled binaries in those implementations?

After checking the manual, it shows that we can retrieve the compiled binaries with `clGetProgramInfo()` after the execution of `clBuildProgram()`. In detail, we need four steps to get the binaries:

1. Allocate the an array of `size_t` to save the size of each binaries.
2. Get the **size** of each binaries with `clGetProgrmaInfo(program, CL_PROGRAM_BINARY_SIZES, ...)`.
3. Allocate the buffers for the binaries. The size of each buffers should be greater than or equal to the size returned in the second step.
4. Get the **binaries** with `clGetProgramInfo(program, CL_PROGRAM_BINARIES, ...)`.

Here's the code listing:

```
cl_int write_binaries(cl_program program, unsigned num_devices,
                      cl_uint platform_idx) {
  unsigned i;
```

```
cl_int err = CL_SUCCESS;
size_t *binaries_size = NULL;
unsigned char **binaries_ptr = NULL;

// Read the binaries size
size_t binaries_size_alloc_size = sizeof(size_t) * num_devices;
binaries_size = (size_t *)malloc(binaries_size_alloc_size);
if (!binaries_size) {
  err = CL_OUT_OF_HOST_MEMORY;
  goto cleanup;
}

err = clGetProgramInfo(program, CL_PROGRAM_BINARY_SIZES,
                       binaries_size_alloc_size, binaries_size, N
if (err != CL_SUCCESS) {
  goto cleanup;
}

// Read the binaries
size_t binaries_ptr_alloc_size = sizeof(unsigned char *) * num_de
binaries_ptr = (unsigned char **)malloc(binaries_ptr_alloc_size);
if (!binaries_ptr) {
  err = CL_OUT_OF_HOST_MEMORY;
  goto cleanup;
}
memset(binaries_ptr, 0, binaries_ptr_alloc_size);
for (i = 0; i < num_devices; ++i) {
  binaries_ptr[i] = (unsigned char *)malloc(binaries_size[i]);
  if (!binaries_ptr[i]) {
    err = CL_OUT_OF_HOST_MEMORY;
    goto cleanup;
  }
}

err = clGetProgramInfo(program, CL_PROGRAM_BINARIES,
                       binaries_ptr_alloc_size,
                       binaries_ptr, NULL);
if (err != CL_SUCCESS) {
  goto cleanup;
```

```
  }

  // Write the binaries to file
  for (i = 0; i < num_devices; ++i) {
    // Create output file name
    char filename[128];
    snprintf(filename, sizeof(filename), "cl-out_%u-%u.bin",
             (unsigned)platform_idx, (unsigned)i);

    // Write the binary to the output file
    write_file(filename, binaries_ptr[i], binaries_size[i]);
  }

cleanup:
  // Free the return value buffer
  if (binaries_ptr) {
    for (i = 0; i < num_devices; ++i) {
      free(binaries_ptr[i]);
    }
    free(binaries_ptr);
  }
  free(binaries_size);

  return err;
}
```

Based on these OpenCL APIs, I have written a simple OpenCL kernel program compiler to translate `.cl` files into pre-compiled binaries. Please refer to cl-compile.c for the source code.

This completes our discussion on the compilation of OpenCL kernel binaries. In the next post, I would like to give an introduction to OpenCL SPIR, the official intermediate representation for OpenCL kernel programs.