Introduction to Massively Parallel Programming on Epiphany

Shinichi Yamagiwa
Associate Professor, Ph.D Engineering
Faculty of Engineering, Information and Systems
University of Tsukuba, JAPAN
Email: yamagiwa@cs.tsukuba.ac.jp
Where is Epiphany on Parallella?

Host processor controls Epiphany by:
(1) Initializing the system
(2) Downloading program
(3) Initializing memory
(4) Writing the input data
(5) Reading the results
(6) Finalizing the system
Inside of Epiphany

Processors are distributed on each cross points of 2D mesh topology.
(Current release supports 4x4 or 8x8)
How to begin Parallella

• Download the latest Ubuntu kernel and write it to SD card.

• Initially, DHCP is configured.

• Connect to LAN, and find IP/MAC address in your router. (MAC is 04:4F:XX:XX:XX:XX)

• Connect terminal such as Teraterm via SSH

• Login by user: parallella pass: parallella

  Strongly recommend to change password by “passwd” command!

  Epiphany SDK is already ready in/opt/keep/epiphany-sdk
How to setup Epiphany SDK

• The latest kernel includes Epiphany SDK (ESDK) and Browndeer COPRTHR SDK.
• Currently available programming methods
  – Using Epiphany SDK, Hardwired programming
  – Using COPRTHR SDK, OpenCL

In the future, MPI will be available

The COPRTHR MPI library is not, and will not be, a part of the currently available COPRTHR-1 SDK. Instead it will be available in the future as part of the new COPRTHR-2 software package being developed by BDT.
Hardwired programming by Epiphany SDK

• Programming tools (mainly used)
  – e-gcc : Compiler for Epiphany architecture.
  – e-server : debugger server.
  – e-reset : reset Epiphany
  – e-hw-rev : Checking Epiphany spec.
  – e-loader : loading program into eCore(s)
  – e-read : read memory contents
  – e-write : write memory contents

  Dispatch program form commandline.
  – e-run : invoke downloaded program on eCore
Epiphany simulator is prepared.
DEMONSTRATION ESDK TOOLS
Program invocation from commandline

- e-gcc
- Program on eCore
- Executable on eCore
- e-loader
- e-write
- Initial data
- e-read
- Result

2015/5/30
Program invocation from host program

Linker information (memory allocation info)

Executable on eCore (Motorola S format)

Excitable on ARM

Loading ELF to eCores

Setting up memory

Invoking ELF

Reading results
Library functions for host program

- Include “e_hal.h” and link “libe-hal.a”.

- Main functions:
  - e_init(), e_finalize():
    Initializing and finalizing Epiphany system.
  - e_reset_system(): reset Epiphany.
  - e_shm_alloc(), e_shm_release(ShmName):
    shared memory allocation and release by name
  - e_shm_attach():
    acquire the shared memory searched by the name
  - e_open(), e_close():
    open and close the Epiphany processor interface
  - e_load(): load the Epiphany program to core(s)
  - e_read(), e_write(): read and write the data on the Epiphany memory.
Library functions for Epiphany program

• Include “e_lib.h” and link “libe-lib.a”.

• Main functions:
  – e_get_coreid(): get processro ID.
  – e_coords_from_coreid():
    get coordinates of the processor in the group.
  – e_shm_attach():
    acquire the shared memory searched by the name
  – e_read(), e_write():
    read and write the data on the Epiphany memory.
Hello World example

• ARM executes an Epiphany program.
• The Epiphany program will write some message to a shared memory.
• ARM reads the message from the shared memory.
Program on host (ARM)

```c
int main(int argc, char *argv[])
{
    unsigned row, col, coreid, i, j;
    e_init(NULL);
    e_reset_system();
    e_get_platform_info(&platform);

    rc = e shm alloc(&mbuf, ShmName, ShmSize);
    if (rc != E_OK)
        rc = e shm attach(&mbuf, ShmName);

    Initializing Epiphany
    Reset Epiphany
    Getting platform information
    Shard memory allocation
    Acquiring the shared memory
    and maps the memory address to the host side.
```
for (i=0; i<platform.rows; i++)
  for (j=0; j<platform.rows; j++)
  {
    e_open(&dev, 0, 0, platform.rows, platform.cols);
    if ( E_OK != e_load("hello-world.srec", &dev, row, col, E_TRUE) ) {
      fprintf(stderr, "Failed to load hello-world.srec\n");
      return EXIT_FAILURE;
    }
    usleep(10000);
    e_read(&mbuf, 0, 0, 0, buf, ShmSize);
    printf("%s\n", buf);
    e_close(&dev);
  }
int main(void) {
    coreid = e_get_coreid();
    e_coords_from_coreid(coreid, &my_row, &my_col);
    if ( E_OK != e_shm_attach(&emem, ShmName) ) {
        return EXIT_FAILURE;
    }
    snprintf(buf, sizeof(buf), Msg, coreid, my_row, my_col);
    e_write((void*)&emem, buf, my_row, my_col, NULL, strlen(buf) + 1);
    return EXIT_SUCCESS;
}
DEMONSTRATION (HELLO WORLD)
OpenCL on Epiphany

- Browndeer COPRTHR SDK supports OpenCL on both ARM and Epiphany
- To use ARM, specify CL_DEVICE_TYPE_CPU
- To use Epiphany, specify CL_DEVICE_TYPE_ACCELERATOR
Application example (2D DCT)

DCT

\[
F(i, j) = c(i)c(j) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \cos\left[\frac{(2x+1)i\pi}{2M}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right]
\]

We can parallelize with respect to \(i\) and \(j\).

IDCT

\[
f(x, y) = \frac{4}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} c(i)c(j)F(i, j) \cos\left[\frac{(2x+1)i\pi}{2M}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right]
\]

We can parallelize with respect to \(x\) and \(y\).
Program on Epiphany (DCT OpenCL)

```c
__kernel void dct2d_kern(
    uint w,
    uint h,
    __global float* F,
    __global float* f)
{
    int i = get_global_id(0);
    int j = get_global_id(1);
    int x, y;
    float ci, cj;
    F[i*h+j] = 0;
    ci = i == 0 ? 1/sqrt(2.0f) : 1.0f;
    cj = j == 0 ? 1/sqrt(2.0f) : 1.0f;
    for(x=0; x<w; x++){
        for(y=0; y<h; y++){
            F[i*h+j] += f[x*h+y] * (cos(((2.0f*x+1)*i*M_PI)/(2.0f*w)) * cos(((2.0f*y+1)*j*M_PI)/(2.0f*h)));
        }
    }
    F[i*h+j] = ci * cj * F[i*h+j];
}
```

\[ F(i,j) = c(i)c(j) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cos \left( \frac{(2x+1)i\pi}{2M} \right) \cos \left( \frac{(2y+1)j\pi}{2N} \right) \]
Program on Epiphany (IDCT OpenCL)

__kernel void idct2d_kern(
    uint w,
    uint h,
    __global float* F,
    __global float* f)
{
    int i = get_global_id(0);
    int j = get_global_id(1);
    int x, y;
    float ci, cj;

    F[i*h+j] = 0;
    ci = i==0 ? 1/sqrt(2.0f) : 1.0f;
    cj = j==0 ? 1/sqrt(2.0f) : 1.0f;
    for(x=0; x<w; x++){
        for(y=0; y<h; y++){
            F[i*h+j] += f[x*h+y] * (cos(((2.0f*x+1.0f)*i*M_PI)/(2.0f*w))*cos(((2.0f*y+1.0f)*j*M_PI)/(2.0f*h)));
        }
    }
    F[i*h+j] = ci*cj*F[i*h+j];
}
int main(int argc, char **argv)
{
    clGetPlatformIDs( 0,0,&nplatforms);
    platforms = (cl_platform_id*)malloc(nplatforms*sizeof(cl_platform_id));
    clGetPlatformIDs( nplatforms, platforms, 0);

    cl_context ctx = clCreateContext(ctxprop,1,&dev,0,0,&err);

    cl_command_queue cmdq = clCreateCommandQueue(ctx,dev,0,&err);

    cl_mem F_buf = clCreateBuffer(ctx,CL_MEM_USE_HOST_PTR, sizeof(float)*IMAGE_WIDTH*IMAGE_HEIGHT,F,&err);
    cl_mem f_buf = clCreateBuffer(ctx,CL_MEM_USE_HOST_PTR, sizeof(float)*IMAGE_WIDTH*IMAGE_HEIGHT,f,&err);

    cl_program prg = clCreateProgramWithSource(ctx,1,(const char**)&src, &src_sz,&err);
    clBuildProgram(prg,1,&dev,0,0,0);
    cl_kernel krn = clCreateKernel(prg,"dct2d_kern",&err);
Program on Host

```c
clSetKernelArg(krn, 0, sizeof(cl_uint), &IMAGE_WIDTH);
clSetKernelArg(krn, 1, sizeof(cl_uint), &IMAGE_HEIGHT);
clSetKernelArg(krn, 2, sizeof(cl_mem), &F_buf);
clSetKernelArg(krn, 3, sizeof(cl_mem), &f_buf);

clEnqueueWriteBuffer(cmdq, f_buf, CL_TRUE, 0,
                      sizeof(float)*IMAGE_WIDTH*IMAGE_HEIGHT, f, 0, 0, 0);
clEnqueueNDRangeKernel(cmdq, krn, 2, 0, gtdsz, ltdsz, 0, 0, &ev[0]);
clEnqueueReadBuffer(cmdq, F_buf, CL_TRUE, 0,
                      sizeof(float)*IMAGE_WIDTH*IMAGE_HEIGHT, F, 0, 0, &ev[1]);

err = clWaitForEvents(2, ev);
}
```
DEMONSTRATION (OPENCL ON EPIPHANY)
Summary

• Epiphany is a manycore accelerator with 2D mesh network.
• Epiphany is controlled by the ARM core on Parallella.
• Supported programming tools
  – Hardwired programming SDK (ESDK)
  – Browndeer OpenCL