Utilizing a Java Based IDE for HPC Education

Introduction:

The Java based IDE called Processing is a good educational tool for programming-beginners, or a good integrated-developmental-environment for visual artists. Here, it is shown that Processing can be used for primary education of high performance computing.

System:

Java based IDE: Processing + **JOCL: Java bindings for OpenCL** + **OpenCL-ready device driver for GPGPU**

Real-time, Interactive, Visual

OpenCL: Open Computer Language.
A parallel computing framework for heterogeneous platforms with multicore CPUs, GPGPUs, DSPs, etc.

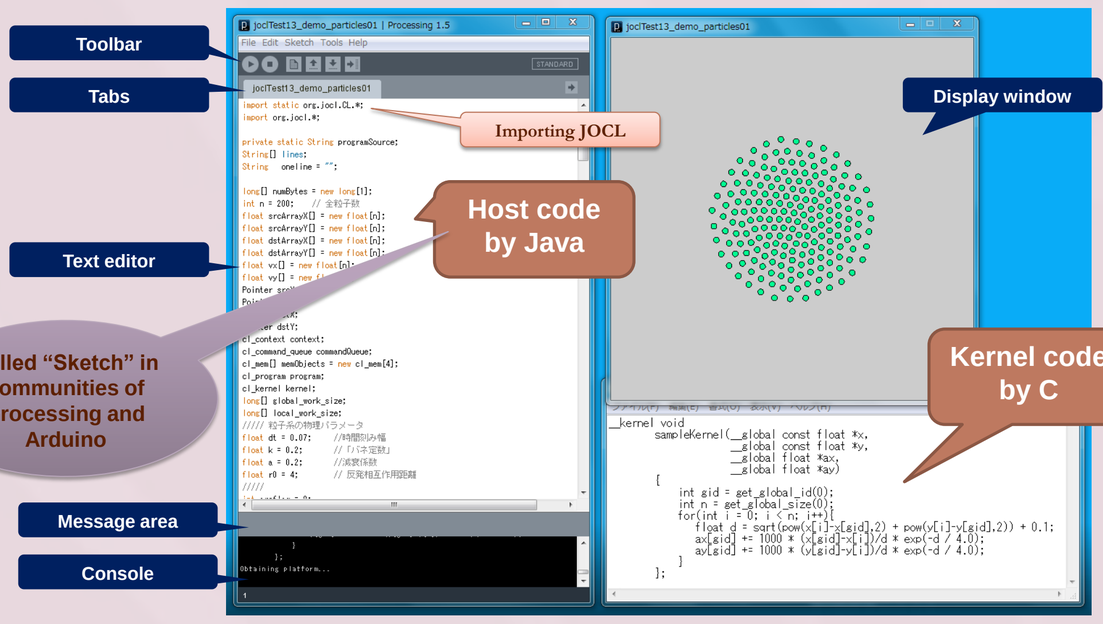
Environment construction

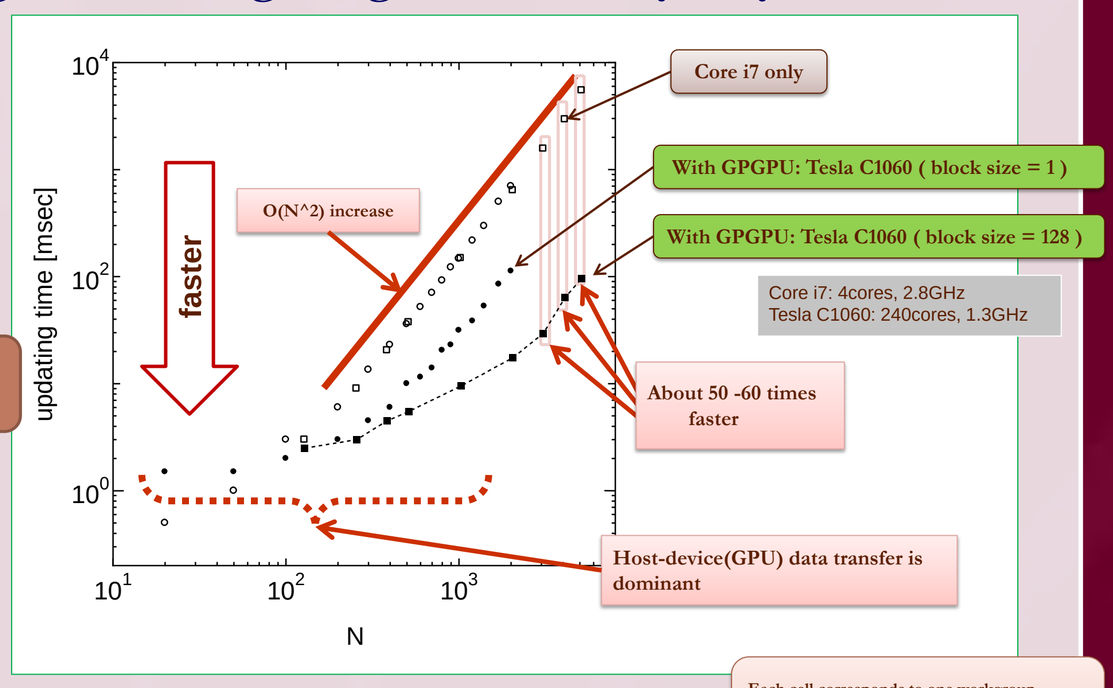
Just put one file: JOCL.jar into an appropriate folder:
<user>/Processing/libraries/jocl/library
and install the GPGPU device driver.
…That's all! Easy!!

Students can get started with GPGPU-computing as HPC quite easily!

Examples:

Case1: N-body problem with a simple O(N^2) algorithm using the global memory only





Case2: Nonlinear partial diffrential equations solved with the local memory

```
/* ********************************************** */
/* ************* sub functions ****************** */
/* ********************************************** */
float _epsilon(int i, int sx, float eps_max){
  return eps_max * (float)i / sx;

}

float _gamma(int j, int sy, float gam_max){
  return gam_max * (1.0 - (float)j / sy);
}
/* ********************************************************************* */
/* ****************** The Kernel function ********************************* */
/* ********************************************************************* */
__kernel void sampleKernel(
                    __global float *u,
                    __global float *v,
                    __global float *un,
                    __global float *vn,
                    __constant float *params,
                    __constant int   *ssparams){

    __local float us[16+2][16+2];
    __local float vs[16+2][16+2];

    int sx = ssparams[0];
    int sy = ssparams[1];

    float dt  = params[0];
    float D_u = params[1];
    float D_v = params[2];
    float epsilon_max = params[3];
    float gamma_max   = params[4];

    int i = get_global_id(0);
    int j = get_global_id(1);

    int is = get_local_id(0) + 1;
    int js = get_local_id(1) + 1;

    us[is][js] = u[i + sx * j];
    vs[is][js] = v[i + sx * j];

    int xp = min(i + 1, sx - 1);
    int xm = max(i - 1, 0);
    int yp = min(j + 1, sy - 1);
    int ym = max(j - 1, 0);

    if(is == 1){
      us[0][js] = u[xm + sx * j];
      vs[0][js] = v[xm + sx * j];
    }
    if(is == 16){
      us[17][js] = u[xp + sx * j];
      vs[17][js] = v[xp + sx * j];
    }
    if(js == 1){
      us[is][0] = u[i + sx * ym];
      vs[is][0] = v[i + sx * ym];
    }
    if(js == 16){
      us[is][17] = u[i + sx * yp];
      vs[is][17] = v[i + sx * yp];
    }

    barrier(CLK_LOCAL_MEM_FENCE);

    float epsilon = _epsilon(i, sx, epsilon_max);
    float gamma = _gamma(j, sy, gamma_max);

    un[i + sx * j] =
      us[is][js] +
      dt * ( us[is][js] * (1.0 - us[is][js]) * (1.0 + us[is][js]) - vs[is][js]) +
      dt * D_u * ((us[is+1][js] + us[is-1][js] + us[is][js+1] + us[is][js-1])/4.0 -
us[is][js]);

    vn[i + sx * j] =
      vs[is][js] +
      dt * epsilon * (us[is][js] - gamma * vs[is][js]) +
      dt * D_v * ((vs[is+1][js] + vs[is-1][js] + vs[is][js+1] + vs[is][js-1])/4.0 -
vs[is][js]);

    u[i + sx * j] = un[i + sx * j];
    v[i + sx * j] = vn[i + sx * j];

}
```

Local memory:

Thread-synchronization in a workgroup

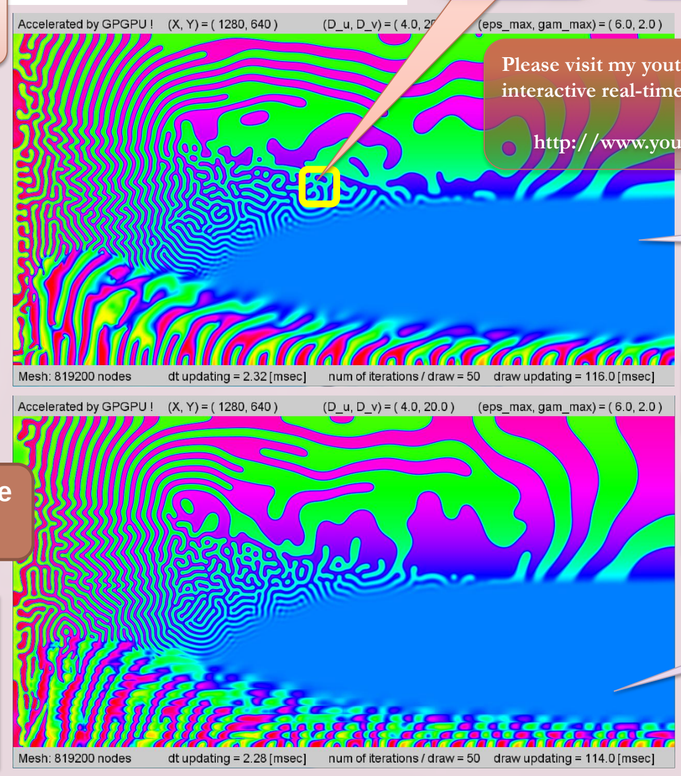
Kernel code by C

A set of partial differential equations

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u + u - u^3 - v$$
$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + \varepsilon(u - \gamma v)$$

Simulation space is divided into many cells.

Each cell corresponds to one workgroup.
One workgroup is composed of 16 x 16 threads, which uses one-unit local memory.



Please visit my youtube channel, in which the interactive real-time simulations are demonstrated.
http://www.youtube.com/user/pftetsuyaGPU

normal

Host-device data transfer in every time step.

Synchronization problem
(host code calls 50 times without host-device data transfer)

anomalous

"Inter-workgroup" de-synchronization occurs. … Why? … The answer is …

Remark:
Processing is a great tool not only for visual arts but also for numerical-simulations.

Future tasks:

Constructing systematic HPC education course.
Preparing the HPC textbook based on utilizing Processing.
Extension to large-scale cluster computing.

Contact: Tetsuya Matsuno at Kyushu University or at Ariake NCT tetsuya@ariake-nct.ac.jp

九州大学 KYUSHU UNIVERSITY

q-CSI Kyushu Univ. Cyber Science Infrastructure

有明工業高等専門学校 Ariake National College of Technology