



Libra Generic Accelerated Computing with Libra SDK

www.gpusystems.com

A short presentation of GPU Systems and introduction to Libra Compute API - Technology Platform and Ecosystem. A technical hands-on "Hello World" tutorial from Libra SDK is presented.

Libra SDK is a highly sophisticated runtime including API, sample programs and documentation for massively accelerating software computations. This introduction tutorial will provide an overview and usage examples of the powerful Libra API & math libraries executing on x86/x64, OpenCL, OpenGL and CUDA technology. Libra API enables generic and portable CPU/GPU computing within software development without the need to create multiple, specific and optimized code paths to support x86, OpenCL, OpenGL or CUDA devices."

Key Libra Facts

- Unleash the Teraflop performance of CPUs and GPUs and future compute accelerators using a single standard programming API to access compute resources.
- Libra API enable heterogeneous computing access through highlevel programming languages, C/C++ and matlab.
- Develop code once, deploy on numerous CPU and GPU devices to execute on a broad range of chip manufacturers.
- Enable more application performance with less code compared to hardware specific low level APIs, easier to maintain applications & algorithms and faster time to product releases – no need to learn new tools or programming languages.

About GPU Systems GPU Systems is a software technology company focusing on enabling the next generation of compute technologies & APIs using current and future hardware architectures. For more information, visit <http://www.gpusystems.com>

Libra SDK examples :

http://www.gpusystems.com/code_examples.aspx

Download Libra SDK :

<http://www.gpusystems.com/download.aspx>

Libra SDK Tutorial 1 :

Libra API to access the power of CUDA, OpenCL/OpenGL and X86/X64 all at once - enabling a powerful CPU/GPU heterogeneous computing platform within a standard C/C++ software programming environment"

Below is a simple "Hello World" Libra C++ application:

Contact: Marco Hjerpe
marco.hjerpe@gpusystems.com
www.gpusystems.com



```

#include <stdio.h>
#include <iostream>
#include <math.h>      // Standard math on x86/x64
#include <Libra.h>     // Standard math on CUDA, OpenCL, OpenGL, x86/x64(multicore)

int main(int argc, char** argv)
{
    // Declare some local variables.
    const int N = 1024;
    long long flopCount;
    double startTime, time, gflops;

    // Declare some more local variables
    // A gVar represents a vector, matrix or scalar
    // with elements of float, double, int or bool
    gVar A, B, C;

    //First we initialize Libra by calling libra_Init().
    if (libra_Init(argc, argv) != 0)
        return 1;

    // Then we set some initial compute states - NOT NECESSARY -
    // Set compute backend to CUDA_, OPENCL_, OPENGL_ or CPU_BACKEND.
    // If no specific backend is specified - Libra Runtime Engine automatically
    // chooses GPU device over CPU device as a default compute backend.
    libra_SetCurrentBackend(CUDA_BACKEND);

    // Set current compute precision to 32bit float
    libra_SetDefaultDataType(GFLOAT32);

    // Create two N by N matrices and initialize all elements to 1.
    A = ones(N, N);
    B = A;

    // initial driver warmups
    C = A * B * 1.23f + 3.15f; C = A * B * 1.23f + 3.15f;

    // We then perform dense matrix multiplication (SGEMM) and time it.
    // For a DGEMM version, set compute precision to GFLOAT64.
    // Start timing
    startTime = libra_GetTime();

    // Perform matrix multiplication, then multiply and bias all elements.
    C = A * B * 1.23f + 3.15f;

    time = libra_GetTime() - startTime;
    printf("Total time : %g%s", time*1000, " milliseconds\n");

    // Compute a performance measurement. Gigaflop / s.
    flopCount = (2*N-1)*N*(long long)N;
    gflops = flopCount / time / 1e9;
    printf("%g%s", gflops, " GFlop/s.\n");

    //After printing measurements our program ends and calls libra_Shutdown()
    //to cleanup system resources.
    libra_Shutdown();

    return 0;
}

```

Contact: Marco Hjerpe
marco.hjerpe@gpusystems.com
www.gpusystems.com

