Comparison on GPGPU frameworks

Torbjörn Sörman 860603-6633
torso632@student.liu.se

Background

Modern graphical processing Units (GPUs) have a lot of computational power. With programmable shaders and floating-point support GPUs was ready for general-purpose computing. Essentially any language that is able to run code on the CPU that poll return values from a GPU shader can create a GPGPU framework.

It is then desirable to utilize the power of the GPU when solving highly parallelizable problems otherwise solved with CPUs. There are several frameworks suitable for general-purpose computing on GPUs (GPGPU), the most important are CUDA, OpenCL, DirectCompute and OpenGL Compute Shaders.

A suitable algorithm to implement is the fast fourier transform (FFT), that have many real world applications. Video encoding and other media features can also be suitable algorithms.

Problem formulation

- What is a suitable algorithm?
- What has been done before within GPGPU framework comparisons?
- How does the frameworks compare in performance? What other considerations are there when choosing a framework (portability, language etc)?
- How does the GPGPU frameworks compare to a modern multicore desktop CPU using OpenMP?

Approach

- Literature study: Find and compare algorithms suitable for GPU implementation.
  - Initially FFT selected as algorithm to implement.
- Literature study: What has been done in the research field of GPGPU comparisons.
- Create a working "Hello world"-example in all frameworks/platforms.
- Learn more about an algorithm and some of the available implementations.
- Get the necessary hardware.
- Implement the algorithm in CUDA and optimize.
- Port implementation to all platforms.
- Run tests and evaluate the results.
- In parallel with implementation, working on the report.

Resources

- CUDA: http://docs.nvidia.com/cuda/
- DirectCompute 11: https://msdn.microsoft.com/en-us/library/ff476330(v=vs.85).aspx

- OpenCL: https://www.khronos.org/opencl/
- OpenGL CS: https://www.opengl.org/wiki/Compute_Shader
- GPGPU: gpgpu.org, wikipedia
- Other: Course material from TDDD56 Multicore and GPU Programming.

Timetable

For full timetable and important events, see separate file.

Early timetable for implementation

Activity	**Comment**	**Time (weeks)**
Planing + Hello World	Successful compilation on all available platforms	2
Planing report		
Literature study		2
Base algorithm	Create a decent implementation to use as base before porting to other platforms	4
Optimizations		
CUDA		
OpenCL		1.5
DirectCompute		1.5
OpenGL Compute Shader		1.5
OpenMP		1.5
Halftime check		
Reserve time		2
Measurements & Evaluation		2
Report	Shall also be written in parallel with implementation and evaluation.	
Opposition		
Presentation		