GPU Accelerated GIMP Plugin Framework

Ajitav Sahoo Prashant Borole Sriram Kashyap

Indian Institute of Technology, Bombay

November 12, 2008

Table of contents

Problem Definition

Problem Statement

Building a framework that enables image processing on parallel graphics hardware thereby dramatically boosting performance. This framework uses OpenGL extensions to implement pixel shaders in GLSL for image processing on GPU.

Motivation

Motivation

- Image processing is inherently a parallelizable task.
- Currently all GIMP filters are implemented as scripts which run on a single CPU. Thus complex image processing tasks are time-consuming.
- The - -enable-mp configure flag works for GIMP application itself, but not for the plugins. So while GIMP can run on multiple CPUs, the plugins/filters do not yet.

Deliverables

- Minimum Guarenteed: A working blur filter that runs on graphics card. ✓

Deliverables

- Minimum Guarenteed: A working blur filter that runs on graphics card. ✓
- A GIMP plugin which runs filters on the graphics card. ✓

Deliverables

- Minimum Guarenteed: A working blur filter that runs on graphics card. ✓
- A GIMP plugin which runs filters on the graphics card. ✓
- A repository of shaders that can be used by the plugin to perform image processing. ✓

Deliverables

- Minimum Guarenteed: A working blur filter that runs on graphics card. ✓
- A GIMP plugin which runs filters on the graphics card. ✓
- A repository of shaders that can be used by the plugin to perform image processing. ✓
- User Framework
  - To enable users to add their own plugins. ✓
  - To configure various shader parameters. ✓

Programming Libraries

- C programming for interface and communicating with GIMP core.
- OpenGL1.2 ARB extensions for image processing on the GPU.
- GLSL programming for implementing pixel shaders.

Libraries Used

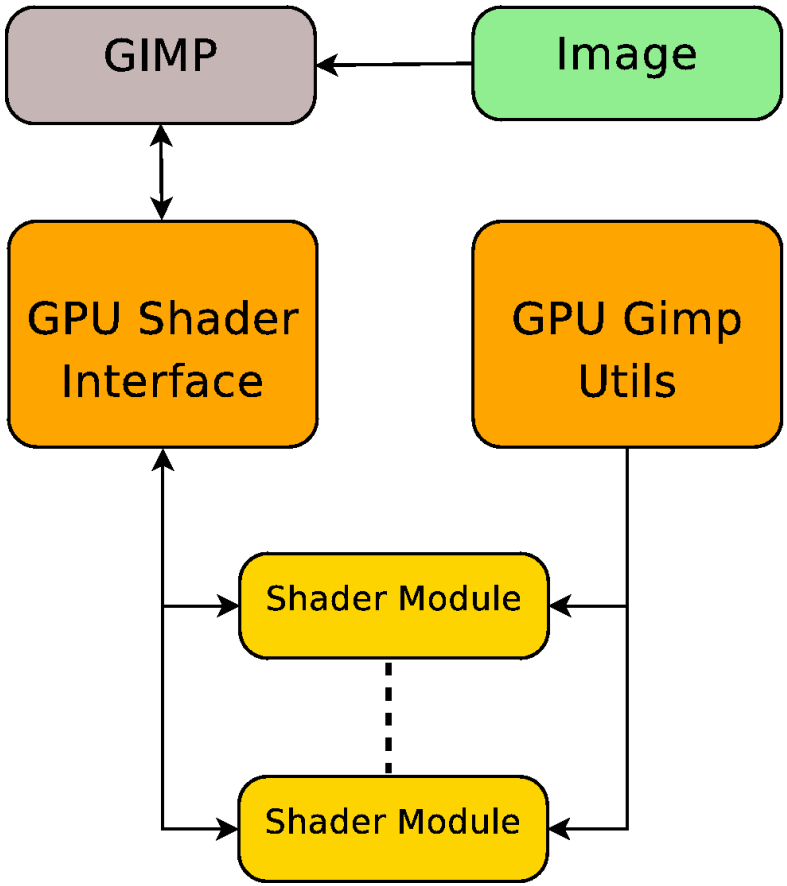
1. GIMP Headers: The developer libraries & headers for GIMP
2. GL: The OpenGL Libraries
3. GLUT: OpenGL Utility Toolkit
4. GLEW: OpenGL Extension Wrangler Library
5. libdl: Dynamic Linking Library

Plan of work

- Getting familiar with GIMP internals.
- Finding ways to execute pixel shaders on GPU.
- Developing pixel shaders.
- Integrating our plugins with the GIMP application.

- We didn't modify any existing GIMP code, but implemented parts of it (using plugins) to run it on parallel hardware.
- Total Time Spent: 50 hours.

System Architecture



Actual Work Done

- Developed a program to transfer buffers between the system memory and GPU memory, i.e. loading a texture to the graphics card.
- Developed a plugin interface to interact with the GIMP core.
- Developed few pixel shader programs to be applied on the image.
- Maintained a library of compiled shaders to be used by GIMP core.

Modules

- The GIMP Shader interface interacts with GIMP as well as the shader modules to provide an abstraction between them.
- The GPU GIMP Utilities are compiled as a single library which provides useful OpenGL functionality to the Shader Modules.
- The Shader Modules execute programs on a graphics card. These programs operate on the image data provided by the GIMP Shader Interface, thus performing image processing on the graphics card.

Acceptance Test & Result

- To check the correctness of the output, we compare the outputs of the existing GIMP filters and the output of our filter.
- We also compare the performance of our filters with the the original filters.

- CPU: Pentium D 2.8 GHz (Family 9 Model 4 Step 7)
- GPU: NVidia 8800 GS (Mem: 128 bit DDR3@1048MHz, GPU@699MHz (G92 A2), Pixel Shader@1728 MHz)

Comparison of CPU vs. GPU times

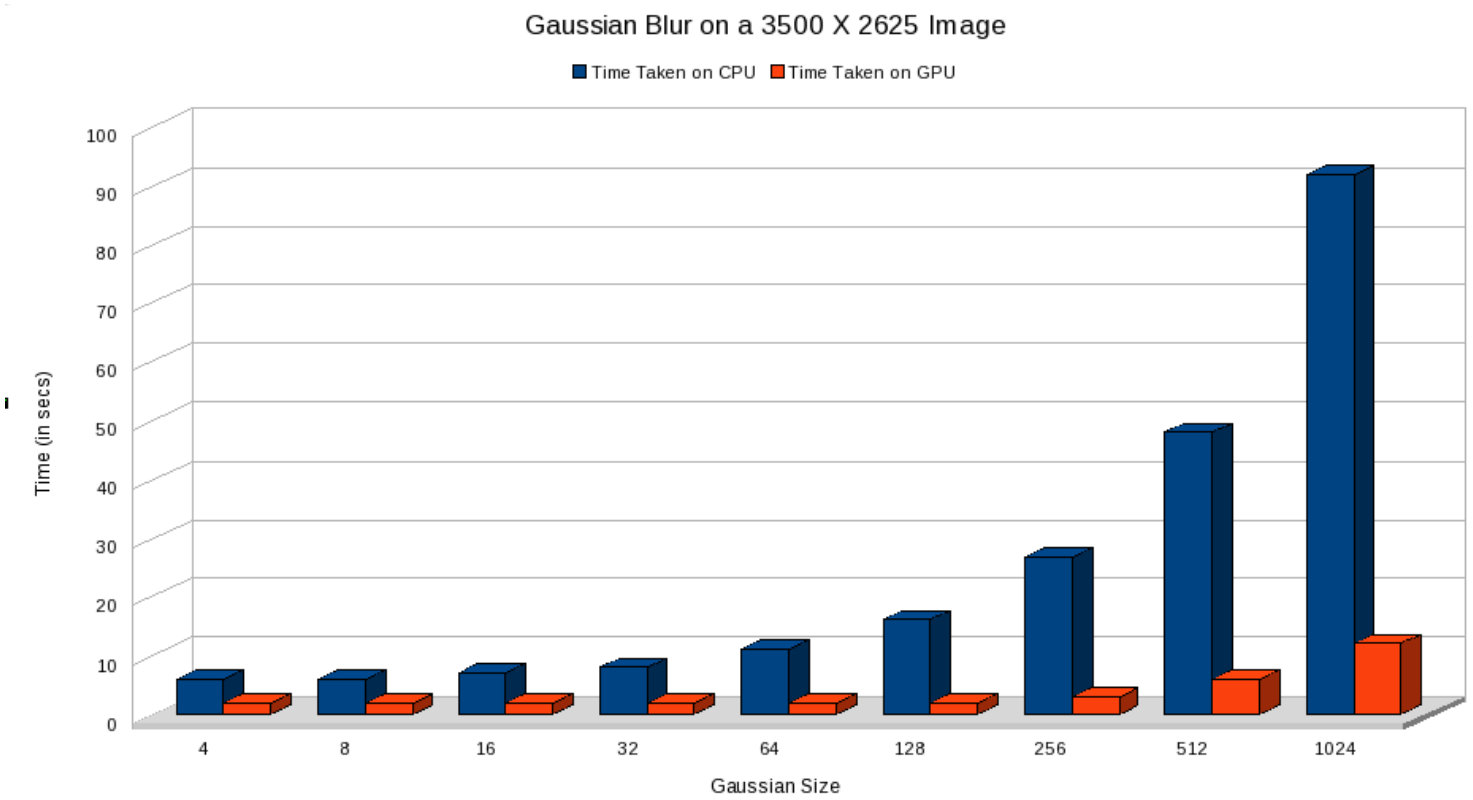


Figure: Comparison of performance on CPU vs. GPU (Gaussian Blur on an image of size 3500 X 2625)

Code execution profile

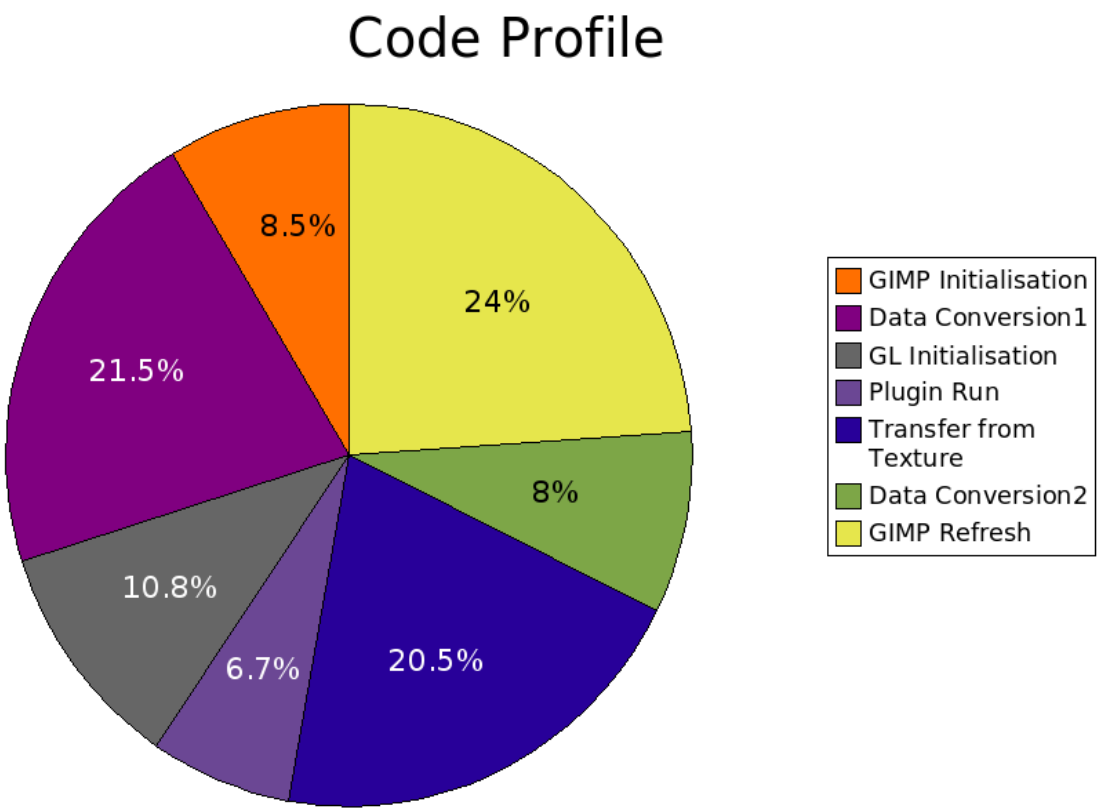


Figure: Comparison of code execution times (for blur: radius 3) within our framework