Science of Computer Programming 143 (2017) 9–19

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Original software publication

cf4ocl: A C framework for OpenCL

Nuno Fachada [a,*], Vitor V. Lopes [b,c], Rui C. Martins [d], Agostinho C. Rosa [a]

[a] Institute for Systems and Robotics (ISR/IST), LARSyS, Instituto Superior Técnico, Av. Rovisco Pais, 1, 1049-001 Lisboa, Portugal
[b] UTEC – Universidad de Ingeniería & Tecnología, Lima, Jr. Medrano Silva 165, Barranco, Lima, Peru
[c] CMAF-CIO, Faculdade de Ciências, Universidade de Lisboa, Campo Grande, 1749-016 Lisboa, Portugal
[d] INESC TEC, Campus da FEUP, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

ARTICLE INFO

Article history:
Received 6 September 2016
Received in revised form 6 February 2017
Accepted 14 March 2017
Available online 23 March 2017

Keywords:
OpenCL
C
GPGPU
High-performance computing
Profiling

ABSTRACT

OpenCL is an open standard for parallel programming of heterogeneous compute devices, such as GPUs, CPUs, DSPs or FPGAs. However, the verbosity of its C host API can hinder application development. In this paper we present cf4ocl, a software library for rapid development of OpenCL programs in pure C. It aims to reduce the verbosity of the OpenCL API, offering straightforward memory management, integrated profiling of events (e.g., kernel execution and data transfers), simple but extensible device selection mechanism and user-friendly error management. We compare two versions of a conceptual application example, one based on cf4ocl, the other developed directly with the OpenCL host API. Results show that the former is simpler to implement and offers more features, at the cost of an effectively negligible computational overhead. Additionally, the tools provided with cf4ocl allowed for a quick analysis on how to optimize the application.

* Corresponding author.
E-mail address: nfachada@laseeb.org (N. Fachada).

http://dx.doi.org/10.1016/j.scico.2017.03.005
0167-6423/

Software metadata

(Executable) Software metadata description	Software metadata
Current software version	2.1.0
Permanent link to executables of this version	https://github.com/ScienceofComputerProgramming/SCICO-D-16-00176
Legal Software License	LGPL-3.0 (library) and GPL-3.0 (utilities)
Computing platform/Operating System	Linux, macOS, Microsoft Windows, BSD, Unix-like
Installation requirements	GLib $\geqslant$ 2.32, OpenCL ICD $\geqslant$ 2.0
User manual	http://www.fakenmc.com/cf4ocl/docs/v2.1.0
Support email for questions	nfachada@laseeb.org

Code metadata

Code metadata description	Code metadata
Current code version	v2.1.0
Permanent link to code/repository used of this code version	https://github.com/ScienceofComputerProgramming/SCICO-D-16-00176
Legal Code License	LGPL-3.0 (library) and GPL-3.0 (utilities, examples, tests, aux. scripts)
Code versioning system used	Git
Software code languages, tools, and services used	C, OpenCL, Bash, Python, CMake
Compilation requirements, operating environments	GLib $\geqslant$ 2.32, OpenCL ICD $\geqslant$ 1.0, CMake $\geqslant$ 2.8.3, C99 compiler, Git (optional), Bash and common GNU utilities (optional)
Developer documentation	http://www.fakenmc.com/cf4ocl/docs/v2.1.0
Support email for questions	nfachada@laseeb.org

1. Introduction

OpenCL is an open standard for parallel programming of heterogeneous compute devices, such as graphics processing units (GPUs), central processing units (CPUs), digital signal processors (DSPs) or field-programmable gate arrays (FPGAs). OpenCL is divided in two parts: a) a C99-based language[1] for programming these devices; and, b) a C application programming interface (API) to launch device programs and manage device memory from a host processor [2].

One of the problems often associated with OpenCL is the verbosity of its C host API [3–5]. Even the simplest of programs requires a considerable amount of repetitive, error-prone boilerplate code. However, the upside is that the API is very flexible and offers fine-grained control of all aspects concerning the development of parallel programs.

In this paper we present the C Framework for OpenCL, cf4ocl, a software library with the following goals: 1) promote the rapid development of OpenCL host programs in C (with support for C++), while avoiding the associated API verbosity; 2) assist in the benchmarking of OpenCL events, such as kernel execution and data transfers; and, 3) simplify the analysis of the OpenCL environment and of kernel requirements. Summarizing, cf4ocl allows the programmer to focus on device code, which is usually more complex and what delivers the end results.

This article is organized as follows. In Section 2, we describe the problem cf4ocl is trying to solve and discuss alternative libraries with similar goals. In Section 3, the architecture and functionality of the library are presented. Implementation details are discussed in Section 4. An example application for massive pseudo-random number generation is introduced in Section 5. Two realizations of this application, implemented in pure OpenCL and using cf4ocl, respectively, are compared and analyzed in Section 6. This paper closes with Section 7, where we provide a number of conclusions regarding the advantages provided by cf4ocl.

2. Problems and background

The OpenCL host API presents a low-level abstraction of the underlying computational platforms. It is well designed, well organized, offering maximum flexibility while supporting a large variety of compute devices. However, this flexibility comes at a cost. As a low-level C API, each function performs a very specific task. Built-in functionality is scarce: there is no error handling or automatic resource management. The end-user must implement this functionality himself. Thus, the simplest of OpenCL host programs requires a considerable amount of repetitive and error-prone boilerplate code.

The problem is minimized when using bindings or wrappers for other programming languages, which abstract away much of the associated complexity. Examples include the official C++ bindings [6], and third-party bindings for languages such as Python [7], Haskell [8], Java [9] or R [10]. In the case of C++, and in spite of the official bindings, the number of wrappers and abstraction libraries is remarkable [11–19]. These libraries aim for a number of goals, such as rapid and/or simplified development of OpenCL programs, high-level abstractions for common computation and communication patterns, embedded OpenCL kernel code within C++ programs or handling of multiple OpenCL platforms and devices.

In turn, there are a number of libraries for developing OpenCL programs in pure C host code. For instance, Simple OpenCL [20] aims to reduce the host code needed to run OpenCL C kernels. It offers a very high-level abstraction, with a minimum

[1] A C++14-based kernel language is introduced in OpenCL 2.1 [1].