OP2 Common C/C++ and Fortran Library Implementation Design Description

Carlo Bertolli and Adam Betts
Imperial College London

December 20, 2015

1 Introduction

This document gives an overview of the OP2 library designed to be used by both C/C++ and Fortran applications. The library has been re-designed starting from C++ and Fortran versions, and its core functions are implemented in C99.

2 Directory Organisation

The main OP2 Common directory contains the following subdirectories:

- **op2**: this directory contains the C/C++ and Fortran OP2 support (see below).
- **apps**: this directory contains the example applications for C++ and Fortran.
- **doc**: this directory contains the generic OP2 Common documentation, as this document.
- **translator**: this directory contains the OP2 source-2-source compilers. It is currently empty, except for the MATLAB compilers, as the translators are developed in different repositories.

3 The OP2 Directory

As specified above, the OP2 directory contains the OP2 support code needed by Fortran and C/C++. The support files are characterised in C/C++ files and in Fortran ones. While the first ones are uniquely implemented in C/C++ and they are self-contained, the Fortran support makes use of the C/C++ one and it is programmed both in Fortran and C.

3.1 The OP2 C Support

The main subdirectories of "op2/c" are the source one ("src") and the include one ("include"). Further directories are "doc" which contains C/C++ OP2 documentation, "lib" and "obj" that are used by the Makefile for the compilation results.

The "src" directory contains:

- The "op_lib_core.cpp" file, which implements the core library routines used to initialise the OP2 data structures, like *op_set* , *op_map* , and *op_dat*. It also defines the lists of actually declared OP2 variables, namely OP_set_list, OP_map_list and OP_dat_list, and related information. The functions provided in this file are called by upper layer functions and they should not be called directly from user programs.
- The "op_reference.c" file provides a set of wrappers to core library functions, to be used in reference implementation. This file does *not* contain the reference implementation of op_par_loop functions.
- The "op_rt_support.c" file implements the OP2 run-time support functions used by any OP2 back-end (i.e. C/C++ and Fortran) that actually needs them. It provides an implementation of the "op_plan_core" function, which builds an execution plan (i.e. based on partitioning and colouring an

OP2 set) based on the input data to an indirect OP2 parallel loop. It also defines the OP2 plan list variable, called "OP_plans". It is worth of noticing that this files also provides an op_rt_exit function which should be called by any OP2 backend using these run-time functions when terminating the OP2 library, via the high-level op_exit.

- A "cuda" directory containing an implementation of high-level OP2 functions properly extending the implementation of the core library. The "op_cuda_decl.c" file provides an implementation of the data structure OP2 declaration functions, which is alternative to the reference one and it must be used by C/C++ and Fortran CUDA back-ends. In particular, the *op_decl_dat* function first calls the op_decl_dat_core function and then it copies the declared data from host to device memory space. The "op_cuda_rt_support.c" file provides an implementation of utility functions wrapping CUDA functions (e.g. cudaMalloc), and a wrapper for the op_plan_core function, which copies part of the generated plan information from host to device memory space. Both files are used by the Fortran CUDA back-end.

- An "openmp" directory contaning the "op_openmp_decl.c" file, which wraps core library functions. This file is used also by the Fortran OpenMP back-end.

The "include" directory contains the following header files:

- op_lib_core.h: this file provides the declaration of the OP2 core library functions (see the corresponding implementation file above).

- op_lib_c.h: this file provide the declaration of the high-level OP2 functions in C. It is included by one of the Fortran wrappers.

- op_lib_cpp.h: this file extends the previous one with C++ OP2 high-level functions, related to the declaration of *op_dat* variables (using templates), constants and global arguments to op_par_loop. The file directly contains the definition of the function, which call the functions declared in the previous header by properly transforming the templated calls to plan C calls.

- op_openmp_rt_support.h: this file declares the op_plan_get function wrapper, which is used by the OpenMP back-ends.

- op_cuda_rt_support.h: this file declares the functions defined in the op_cuda_rt_support.c file (see above).

- op_cuda_reduction.h: this file provides a templated implementation of the op_reduction routines for CUDA back-ends. It is not part of the op_cuda_rt_support.h file (as it was in previous versions) because the previous file is included by a source file that is compiled with a plain C compiler, i.e. not supporting templates. This design is a consequence of the choice of avoiding to label C functions using the notation "extern C" to be called by both Fortran and C++.

It is to be remarked that the C++ and Fortran support might compile the same files with different compilers (e.g. nvcc and gcc). For this reason, some of the include file directly include low level CUDA headers like:

```
#include <cuda.h>
#include <cuda_runtime.h>
#include <cuda_runtime_api.h>
#include <device_launch_parameters.h>
#include <device_functions.h>
```

Figure 1: Example of declaration of op_set variables.

3.1.1 Compiling OP2 C Support

The "op2/c" directory currently contains a Makefile to build proper libraries to be used by OP2 applications. The invocation of the "make" command in the directory will (hopefully) produce the following libraries:

- **libop2_reference.a**: this is the library providing the reference implementation of OP2 calls. It depends on the core library functions and on the reference version-related source files (see above).
- **libop2_rt_support.a**: this is the OP2 run-time support library, and it includes a compilation of the op_rt_support.c file (see above), i.e. an implementation of the op_plan_core function and associated functions and variables.
- **libop2_cuda.a**: this is the library providing an implementation of OP2 functions for CUDA back-ends.
- **libop2_openmp.a**: this is the library providing an implementation of OP2 functions for OpenMP back-ends.

For example, to link an application against the CUDA library, it is sufficient to specify in the g++ command line the "-lop2_cuda" option (i.e. there is no need of using the prefix "lib" and the extension ".a".

4 The Applications Directory

The application directory ("apps") contains examples in Fortran and C++.

4.1 The C+ Application Directory

This directory contains the airfoil and jacobi examples. The airfoil application makes use of the OP2 C common library. To compile it, it is required to set the OP2 environment variable to the path of the OP2 Common "op2" directory (i.e. the one containing the C/C++ and Fortran support directories). For CUDA, it is required to set the CUDA_INSTALL_PATH variables to the path of the local CUDA installation. The compilation proceeds in the following way:

- The sequential implementation is compiled with g++ and linked against the op2_reference library.
- The OpenMP implementation is compiled with g++ and linked against the OP2 run-time library and the openmp one. The run-time library is needed because the OpenMP back-end makes use of the OP2 function to build OP2 plans.
- The CUDA implementation is compiled with nvcc and g++ and linked against the OP2 run-time library (again because the back-end uses the plan function) and the OP2 CUDA library.