H2020 FETHPC-1-2014

Enabling Exascale Fluid Dynamics Simulations
Project Number 671571

D2.5 – Release of ExaFLOW CFD algorithms.

WP2: Efficiency improvements towards exascale.

Document Information

Deliverable Number	D2.5
Deliverable Name	Release of the ExaFLOW algorithms
Due Date	30/09/2018 (PM36)
Deliverable Lead	UEDIN
Authors	N. Johnson, UEDIN M. Bareford, UEDIN N. Offermans, KTH A. Peplinski, KTH N. Jansson, KTH J. Gong, KTH C. Jacobs, SOTON S. Jammy, SOTON D. Moxey, UNEXE/IMPERIAL M. Vymazal, IMPERIAL C. Cantwell, IMPERIAL A. Nielsen, EPFL P. Vogler, USTUTT J. Zhang, USTUTT
Responsible Author	N. Johnson, UEDIN, n.johnson@epcc.ed.ac.uk
Keywords	Software release; algorithm implementation; open source
WP	WP2
Nature	R
Dissemination Level	PU
Final Version Date	26/09/2018
Reviewed by	M. Bareford, UEDIN N. Jansson, KTH
MGT Board Approval	30/09/2018

Document History

Partner	Date	Comments	Version
UEDIN	12/09/2018	Pass 1; basic listing	0.1
UEDIN	17/09/2018	Add small text; group to apps	0.2
UEDIN	20/09/2018	Add remaining links to code	0.3
UEDIN	25/09/2018	Add explicit links to AMG setups	0.4
UEDIN	26/09/2018	Fix comments from reviewers	0.5
KTH	30/09/2018	Final version	1.0

Executive Summary

This deliverable reports that the ExaFLOW algorithms have been developed and released by including them in either the co-design applications, as libraries or modules that can be used with the co-design applications, or as separate codes.

The descriptions of the algorithms were discussed in D1.2 and D1.3 and the implementations and performance are discussed in D2.4.

Contents

1 Introduction

This short deliverable serves to provide a set of pointers to the ExaFLOW CFD algorithms as implemented in the various applications, both co-design and stand-alone, used in the project.

During the ExaFLOW project, we chose to release algorithmic implementations as improvements to applications, with the license of the surrounding application, hence all improvements are open source, with a variety of licenses.

In this document, the algorithms are grouped by the application in which their implementation appears.

2 Implementations of Algorithms

2.1 Nektar++

Nektar++ is one of our co-design applications. It is a high order spectral element code written primarily in C++ for work on CFD problems. It is primarily maintained by the team at IMPERIAL but accepts contributions from other users and developers. It uses the MIT license to permit commercial usage of the code and allows companies and other research groups to copy, adapt and release their own versions.

Two primary contributions have been made to Nektar++ by ExaFLOW, the first is improvement works to the IO subsystem, including work to assist in profiling IO performance and the second is a basic working implementation of the ULFM fault tolerance mechanism. They can be found in the two branches linked below:

IO
https://gitlab.nektar.info/nektar/nektar/commits/feature/ioext
ULFM
https://gitlab.nektar.info/nektar/nektar/commits/feature/ulfm

Installation and use of the ExaFLOW contributions is by building the code from the relevant branches. The ioext branch is simply a normal branch of the Nektar++ application with the improvements to IO rolled-in. Usage of the fault-tolerance work in the ULFM branch is covered in detail in deliverable D2.4, section 2.6.

2.2 Nek5000

Nek5000 is another of two co-design applications. It is a high-order solver code written primarily in Fortran, C and pure MPI for work on CFD problems. It is primarily maintained by Argonne National Laboratory (ANL) in the USA with many contributions from the team at KTH in Stockholm. ExaFLOW has contributed to the Adaptive Mesh Refinement (AMR) code and Algebraic Multi-

Grid (AMG) codes, the latter being replaced in many places with calls to the Hypre library for further performance gains. Their inclusions can be found at the links below. KTH maintains a repository of pre-release code under active development which is ahead of the main repository. We link to the KTH versions as ExaFLOW contributions are more common here. In time, these will be adopted into the main repository hosted by ANL. Explicit setup and usage instructions are found in the documentation in the repositories themselves, this approach keeps the documentation tied to the specific version of the code available.

AMR
In the nekp4est branch awaiting merging to the main branch.
https://github.com/KTH-ExaFlow/Nek5000.git

AMG
Available in the main branch as of commit with hash ending: c6a20a
https://github.com/KTH-ExaFlow/Nek5000.git

Further, the standalone serial setup for AMG is now found in the main (ANL) repository:
https://github.com/Nek5000/Nek5000/blob/master/tools/amg_setup/amg_setup.c

ExaFLOW has also developed a parallel version of both the AMG setup and the AMG solver, which are directly included in the Nek5000 code. The main file resides in the developers repository, awaiting promotion to the KTH repository:
https://github.com/nicooff/nek5000/blob/amg_hypre/core/experimental/crs_hypre.f

GPU
GPU support is not present in the main Nek5000 code, as it sticks to a CPU implementation. The extension to GPUs is performed by using OpenACC directives allowing the code to be agnostic of which brand of GPU, or other accelerator, is offered in any given system. The OpenACC port of Nek5000 can be found at the link below:
https://github.com/Nek5000/Nek5000/tree/openacc

2.3 OpenSBLI

SBLI was a third co-design application at the outset of the project but has now been replaced by the open-source OpenSBLI code. Some early project results and use-cases used SBLI. OpenSBLI uses code generation to turn a system of equations coded in a domain specific language (DSL) into code for the target system. The code generator is very flexible and allows for a variety of target operating environments such as CPU or GPU based machines. The OpenSBLI code can be found at the link below.
https://github.com/opensbli/opensbli

2.4 Other libraries and applications

ExaGS is a fork of the GS library originally supplied with Nek5000. Whilst GS uses pure MPI for communication, the goal of ExaGS was to use UPC to test the performance gains of using single sided communications, and to further help separate the communication library from the main code and improve the API documentation to make porting to other languages and frameworks such as OpenSHMEM or ZeroMQ simpler. It can be found at the link below.
https://github.com/EPCCed/exaflow-exags-upc

The work on compression algorithms has been implemented in an in-house variation of the NS3D code developed by USTUTT. It is not currently available outside that partner but there is a plan to release it as open-source software by the end of 2018.

3 Conclusion

This deliverable reported on the dissemination of the ExaFLOW algorithms by their inclusion into open source (or soon to be open source) applications and libraries to allow use beyond the ExaFLOW project.