Para 2010 – State of the Art in Scientific and Parallel Computing – extended abstract no. 44
University of Iceland, Reykjavik, June 6–9 2010
http://vefir.hi.is/para10/extab/para10-paper-44.pdf

The FlowSimulator framework for massively parallel CFD applications

Michael Meinel*[1] and Gunnar Ólafur Einarsson†[2]

[1]*German Aerospace Center (DLR), Simulation and Software Technology*
[2]*German Aerospace Center (DLR), Center for Computer Applications in Aerospace Science and Engineering*

Abstract In this paper the FlowSimulator framework for multi-disciplinary computational fluid dynamics simulations on high performance computer platforms is described. An overview of the provided functionality is given and possible benefits for tool developers as well as design engineers are presented.

Keywords CFD, HPC, multi-disciplinary framework, coupled simulation

1 Introduction

Over the past several years the field of computational fluid dynamics (CFD) evolved from being a stand alone discipline to being a part of a complex multi-disciplinary simulation task. Design engineers long to integrate a growing number of tools into their production workflows as the precise prediction of an aircraft's flight performance requires the simulation of the whole flight envelope.

1.1 Evolution of CFD workflows

When computational fluid dynamics (CFD) came on in the early 1970's, it was a great challenge on its own. This new combination of physics, numerical mathematics and computer science was supported by the availability of high performance computer platforms [3].

Although mono-discipline CFD research has reached a relatively mature state, and the use of CFD in the aircraft industry is increasing, physical modelling and algorithm development still remain as challenging research topics as ever. The reduction of the computational time needed by CFD software is one of the driving goals of current research activities, especially when considered from a financial point of view. Due to the increased acceptance of CFD in industrial process chains, the desire to combine CFD with other disciplines in order to improve the overall design of a given product has awakened. Thus, increasing effort is being put into coupled simulations of the different disciplines that affect aviation systems [5].

As an example of the need for multi-disciplinary simulations is the interaction between the aerodynamic forces acting on the aircraft structure, and the deflection of the structure due to those forces. The changes in the shape of the aircraft have an impact on the aerodynamic performance, and this information is something that a design engineer would like to have available. Consequently, those two applications, CFD and computational structural mechanics (CSM) have to be coupled "in-the-loop", first, to determine the steady aeroelastic equilibrium, and second, to determine the response of the whole system during a flight maneuver. However, in order to simulate complete maneuvers, an additional flight mechanics module has to be incorporated into the workflow as well [9]. New disciplines are also being evolved, like computational aeroacoustics (CAA), which need very accurate aerodynamic solutions to use as starting points to determine acoustic sources [6].

1.2 Incorporation of common tools

The classical CFD workflow usually consists of four steps in order to get from the inital geometry description to a solution of the flow field, as depicted in Figure 1:

- geometry design;
- mesh generation;
- evaluating the actual flow solution and
- postprocessing to extract the required information.

Starting point for any simulation is the geometry which is usually the result of a computer aided design (CAD) process. These parametrical curves and surfaces are then passed to a mesh generation program which tesselates the surfaces and creates a spacial discretization.

Before the actual CFD solver can work on this numerical domain, a preprocessing step is required to determine some extra data such as the size of time steps to simulate or invariant derived data as wall normals or local volumina. Although some CFD codes integrate this preprocessing step, it is not unusual to have it as an extra tool.

With the preprocessed data, the actual CFD solver is able to calculate a flow field solution for the problem given.

*Email: michael.meinel@dlr.de
†Email: gunnar.einarsson@dlr.de

Figure 1: Classical workflow of CFD applications.

However, the output has to be further postprocessed to extract the relevant data such as integral values and to format them in a way that can be displayed by one of several available visualization tools.

1.3 New challenges in aerodynamic simulations

In the classical workflow, the most computationally intensive part is the CFD solver itself. The impact of the mesh generation, pre- and postprocessing are usually negligible. Thus the common practice for steady flows is to have the tools exchange data using file in- and output. As different tools and different vendors often use their own data formats, each step may require the inclusion of format converters in the workflow.

With the development of more complex workflows such as depicted in Figure 2 the dominant operation shifted from one tool (the CFD solver) to a combination of tools (CFD, CSM solvers and flight mechanics in this case). Using the classical file-and-converter-strategy for communication between the tools may lead to a significant performance drop, as these operations now have to be done "in-the-loop" and may thus become, especially for massively parallel applications, a performance drag.

Another observable effect is closely related to the use of recently available parallel network file systems for cluster computers: When some thousands of processes try to access the file system at the same time, they produce a large amount of network traffic which slows down the overall performance of the cluster and might even bring the complete file system down. This depends, of course, a lot on how much of the cluster hardware is dedicated to file system handling instead of computational resources.

2 FlowSimulator software architecture

The FlowSimulator software, as initiated by Airbus, EADS[1] Military Air Systems, and several European Research Institutes, is designed to be a platform where multi-disciplinary simulations can be run on massively parallel cluster architectures as efficiently as possible. The software is intended to address the current and future needs of design engineers who require multi-disciplinary analysis in their work. The FlowSimulator platform establishes a software architecture as sketched in Figure 3 [4].

[1]The European Aearonautic Defence and Space Company (EADS) is the biggest company in Europe in the fields of aeronautics, space and defence technology.

Figure 2: Sample workflow for coupled CFD simulation.

2.1 Python as integration environment

In order to provide easy extensibility of the FlowSimulator, a plugin system has been implemented. Instead of reinventing the wheel, Python was chosen as the integration environment [8]. Although this approach requires that the respective tools might have to be slightly modified in order to allow a connection to Python, the assumption is that the benefits far outweigh the effort required for such modifications.

Over the past several years, Python has gained a lot of acceptance within the scientific community, not only due to great packages like NumPy and SciPy [2], but also for the ease with which it can be used as the "glue" between diverse software components. With these packages and qualities, Python allows for the rapid development of pro-

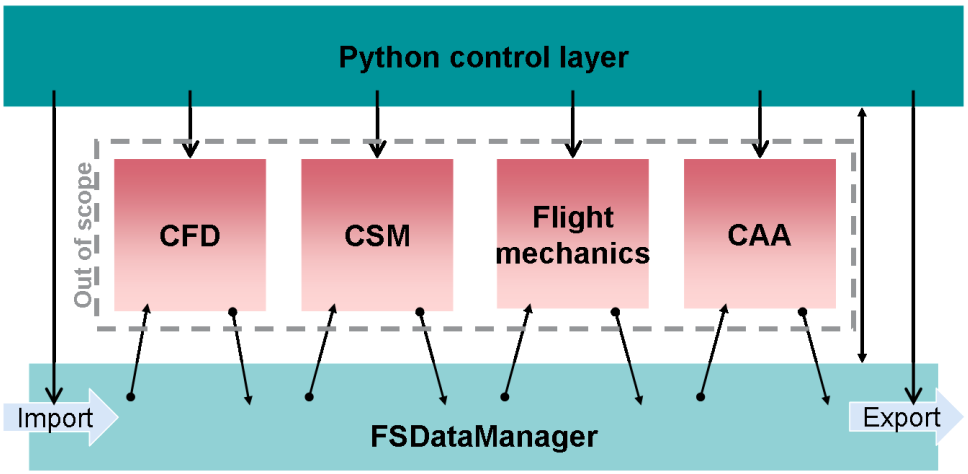

Figure 3: Basic architecture of the FlowSimulator framework.

totypes which integrate almost ad-hoc into the FlowSimulator environment.

Thanks to tools like SWIG[2] [10] and f2py[3], legacy codes are easily converted to Python modules which can then be loaded at runtime, thus providing a dynamic plug-in infrastructure.

2.2 An open framework instead of a middleware library

In order to more easily reach a wide range of tool developers, an extensive open interface to the FlowSimulator DataManager (FSDM) is provided, instead of using a common library which would have to be integrated in each tool. The interface is realized as a collection of C++ classes that are all wrapped to Python, leaving the choice open to the tool designer of how to best integrate their code with the FSDM: through the Python layer of FSDM, or directly with the C++ classes.

The open interface approach of integrating a tool with the framework should lead to a reduction in the cost of the integration, since extensive modifications of the existing code are not required. In general, only a new import and export module that interfaces with the FSDM has to be implemented. As a consequence, the exchange of data between one tool and another can now take place completely without file I/O, since all tools can read and write directly to the FSDM.

Another positive effect of the integration with the FSDM is that the import and export filter of one tool require no information on the data format required by any of the other tools being used in a given workflow. Since each tool imports and exports data directly to the FSDM, the data transfer between the tools is completely independent of the internal data format used by the individual tools. This independence can be easily exploited to create a workflow using tools that previously had no common interface through which they could share data.

Yet another positive aspect of the integration with the FSDM is that the integration can be done without linking the tool against a proprietary library, thus avoiding any type of licensing costs. Moreover, since any given tool is treated as a plugin by the FSDM and the core of the FlowSimulator does not depend on any of the plugins, the question of availability and licensing for individual plugins is completely at the discretion of the plugin owner.[4]

2.3 A massive parallel environment

Current high performance computers are mainly built as distributed memory systems with some kind of cluster architecture. Thus, modern high performance applications need to be parallelized in order to use the resources of these achitectures as efficiently as possible. Although partitioned global address space (PGAS) systems are emerging, the most common backend for data-parallel applications is still the Message Passing Interface (MPI) [7].

In order to provide an efficient environment for parallel applications, the FlowSimulator framework is designed to operate natively on massively parallel cluster architectures. The FSDM, as the backbone of the FlowSimulator, is implemented to take full advantage of a parallel environment. Every data structure defined in the base library knows its own communication context and provides convenient methods to easily (re-)distribute the data in a non-blocking manner.

This built-in parallelism of the FSDM supports the parallel tools in a given CFD workflow to access data without the need of a central coordination instance. The redistribution of the data according to the needs of the tools can be easily achieved within the FSDM and is optimized for the

[2] SWIG is a software development tool that connects programs written in C and C++ with a variety of high-level programming languages.

[3] f2py is part of the NumPy distribution and allows automatic integration of Fortran codes with Python.

[4] Although provided at no cost to project partners, the FSDM is not provided under an open source license.

layout of the provided data structures.

2.4 Included base functionality

FSDM comes with some basic tools already built in. Most important of these is a wide range of import and export filters that can handle most of the common file formats used in CFD such as CGNS[1], NetCDF[12] and Tecplot[11]. As with all FSDM modules, the import and export filters are fully parallel from the ground up, allowing large datasets to be handled with ease, as there is no need to store all of the data in the memory of a single compute node at any time.

Another important functionality closely related to the import and export filters are the built in partitioning algorithms. These allow the distribution of the data independent from the tools that are to use them and allows a domain decomposition that is invariant from the tool to be used.

Other tools, like some geometry operations (transformation and basic tesselation), as well as a basic linear algebra module are provided in addition.

3 Conclusion

The FlowSimulator provides an open framework that can be used by tool developers for integrating their software into a common environment that has been designed to enhance CFD related, multi-disciplinary numerical simulations on massively parallel high-performance computer architectures. The plugin enabled framework allows design engineers to select the tools they wish to integrate into their specific multi-disciplinary workflow chain, treating each tool as a black box, since all data transfer and conversion is taken care of by the FlowSimulator.

This does not only allow for easy combination of tools into workflows but also provides the opportunity to easily replace a tool or to have a fair comparison between different tools as the environment is consistent.

As the import and export of files is managed by the FlowSimulator DataManager, a tool that is integrated into the framework can use virtually every file format already integrated in the FSDM.

The list of tools already integrated in the FlowSimulator contains most of the tools used in current production environments, and the list is constantly growing. Some examples of integrated tools are

- import and export filters for the most common geometry and mesh formats;
- a mesh generator;
- structured and unstructured CFD solvers;
- a structural mechanics code;
- a flight mechanics module;
- different pre- and postprocessors and
- visualization tools for on-line visualization of the simulation results.

Modules which are expected to be integrated in the near future include

- aeroacoustics codes and
- special domain tools like buffeting simulations.

The FlowSimulator has already been established at industrial partners, and first production workflows on high performance computers have been implemented.

References

[1] CFD General Notation System. `http://cgns.sourceforge.net`.

[2] NumPy / SciPy homepage. `http://scipy.org`.

[3] J. Blazek. *Computational Fluid Dynamics: Principles and Applications*. Elsevier Ltd., second edition, 2005.

[4] H. Bleeke (Airbus) and U. Tremel (EADS MAS). FlowSimulator Review Meeting. November 2006.

[5] S. Cant. High-performance compuing in computational fluid dynamics: progress and challenges. *Philosophical Transactions of the Royal Society A Mathematical, Physical & Engineering Sciences*, 360:1211–1225, 2002.

[6] R. Ewert. Die Simulation von Strömungslärm bei Verkehrsflügzeugen. In *Fortschritte der Akustik*, volume I, pages 21–22, 2006.

[7] MPI-Forum. Message Passing Interface. `http://www.mpi-forum.org/docs`.

[8] Python. Python website. `http://www.python.org`.

[9] A. Schütte, G. Einarsson, A. Raichle, B. Schöning, M. Orlt, J. Neumann, J. Arnold, W. Mönnich, and T. Forkert. Numerical simulation of maneuvering aircraft by aerodynamic, flight mechanics and structural mechanics coupling. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, volume AIAA 2007-1070, Reno, Nevada, January 2007.

[10] SWIG. Simplified Wrapper and Interface Generator homepage. `http://swig.org`.

[11] Tecplot. Plotting and Data Visualization Software. `http://www.tecplot.com`

[12] Unidata. NetCDF. `http://www.unidata.ucar.edu/software/netcdf`.