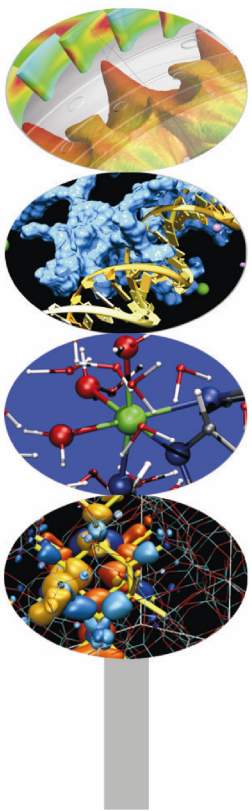
CINECA SCAI
SuperComputing Applications and Innovation

CINECA COURSES High Performance Computing 2014

HPC enabling of OpenFOAM® for CFD applications



Introduction

26-28 March 2014, Casalecchio di Reno, BOLOGNA.

SuperComputing Applications and Innovation Department, CINECA

PRACE CINECA

Table of Contents

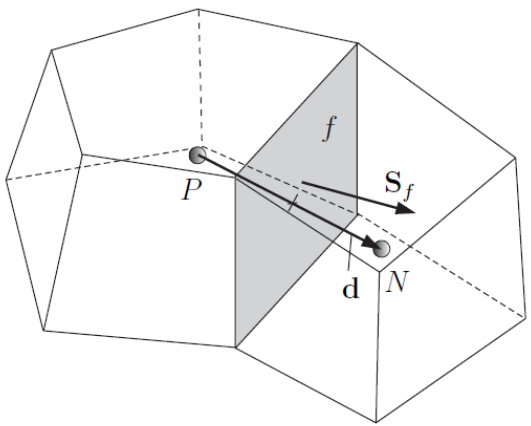
Objectives and Topics

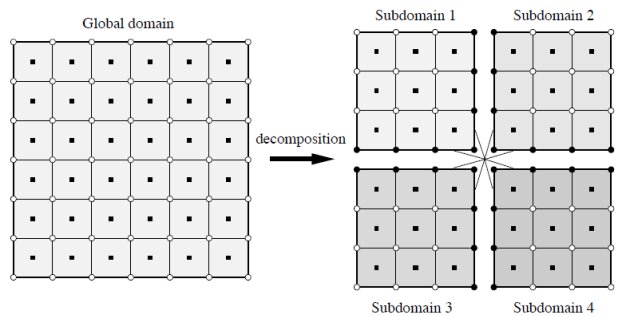
- Objective
  Give an overview of OpenFOAM® (**Open** **F**ield **O**peration **A**nd **M**anipulation) CFD Toolbox, an open source object oriented CFD simulation platform based on fundamental ideas of object orientation, layer software design and equation mimicking.
- Topics
  - What is OpenFOAM
  - Open Source Code
  - Equation mimicking
  - Object Orientation
  - Top-Level Solver Structure: Applications ans solvers
  - OpenFOAM @ CINECA
  - Parallel Computing
  - Remarks: Pro and cons

Table of Contents

What is OpenFOAM

The OpenFOAM® (Open Field Operation and Manipulation) CFD Toolbox is a free, open source CFD software package which has a large user base across most areas of engineering and science, from both commercial and academic organisations.
OpenFOAM has an extensive range of features to solve anything from complex fluid flows involving chemical reactions, turbulence and heat transfer, to solid dynamics and electromagnetics. It includes tools for meshing, snappyHexMesh, a parallelised mesher, and for pre- and post-processing. OpenFOAM is written in highly efficient C++ object-oriented language programme.
It uses finite volume approach to solve system of PDE (Partial Differential Equations) ascribed in any 3D unstructured mesh of polyhedral cells. Almost everything (including meshing, and pre- and post-processing) runs in parallel as standard.
Domain decomposition parallelism is integrated at low level, thus enabling users to take full advantage of computer hardware at their disposal.





OpenFOAM history

- Over the past decade, the level of sophistication and quality of open-source software has significantly grown, largely aided by the move to *object-oriented programming* and on-line version-control repositories (e.g. SourceForge, GitHub).
- OpenFOAM born in the strong CFD British tradition, specifically at the Imperial College, London.
- The original development of OpenFOAM was begun by Prof. David Gosman and Dr. Radd Issa, with principal developers Henry Weller and Dr. Hrvoje Jasak.
- The main idea was to develop a code based on *finite Volume Method*, using C++ and *object-oriented* programming to develop a syntactical model of *equation mimicking* and scalar-vector-tensor operations.
- A large number of Ph.D. students and their theses have contributed to the project.
- Weller and Jasak founded the company Nabla Ltd., but it was not successful in marketing its product, FOAM (the predecessor of OpenFOAM) and the company folded in 2004.
- In 2004 Weller founded the *OpenCFD Ltd.* and released the GNU general public license of OpenFOAM software.
- in August 2011 *OpenCFD was acquired by Silicon Graphics International (SGI)*.
- In September 2012, SGI sold OpenCFD Ltd. to the *ESI Group*.

What is OpenFOAM

The "Engine" of OF: the Numerical Method

To solve equations for a continuum, OpenFOAM uses a numerical approach with the following features:

- Segregated, iterative solution: For the system of equations governing our problem of interest, separate matrix equations are created for each equation, and are solved within an iterative sequence (as opposed to created one, big matrix equation for the entire system of equations).
- Finite volume method: Matrix equations are constructed using the finite volume method applied to arbitrary shaped cells (any number of faces, any number of edges).
- Co-located variables: The solution variable for each matrix equation is defined at cell centres.
- Equation coupling: The coupling between equations, particularly pressure and velocity is performed using adapted versions of well-known algorithms such as e.g. PISO and SIMPLE.

Table of Contents

Open Source Code

- Open Source: the software is distributed as source code
  - **OpenFOAM-2.3.0** distributed by ESI, last release 17^{th} February 2014.
  - OpenFOAM-ext-3.0 "Jeju" distributed by Extend Project Community. Released via git as OpenFOAM extension
- No license fees and you are free to modify the software
- OpenFOAM is licensed under the GNU GPL (General Public License).
  If GPL software is redistributed, the source code must be made available
- Why Open Source?
  Academic rationale for open source code is clear: open collaboration and sharing. Industrial users rely on commercial software with strict quality control and dedicated support teams, but its flexibility is not sufficient, development is too slow, support quality varies and parallel CFD licences has to paid
- Open Source is a software feature
  - Reminder: Open Source and GPL does not imply zero price
  - Computer time has to be paid, but cost is unavoidable
  - Software support, help with running and customisation is still required
  - Engineers and CFD analysts running the code are the most costly part: **better**!

Open Source Code

- Mode of operation:
  When a CFD code works well in a design process, it will be used in large volume. Development and validation may need to be funded by user but further cost drops significantly: no annual license fee to pay. Parts of acceptance and validation effort become responsibility of the user
- Users and system integrators of OpenFOAM can
  - modify the software freely
  - asses code quality
  - know what the software is doing
  - deploy OpenFOAM whatever they wish: laptop, own cluster, third party cluster, cloud...
  - influence development priorities
  - manage costs: choose when and how to get something for their money
- F.A.Q. Why ESI has acquired OpenCFD?
  - The virtual engineering market continues to show an increase in the demand for open source software, specifically in the CFD domain
  - ESI believes in the openness of OpenFOAM and will be investing in OpenFOAM to scale it to reach more users

R. Bouwman, "OpenFOAM and ESI: unleashing the power of open source for CFD in HPC simulations", Workshop "HPC enabling of OenFOAM for CFD applications", 26/11/2012, CINECA, Casalecchio di Reno, Italy

Table of Contents

Equation mimicking

- OpenFOAM is first and foremost a *C++ library* used to solve in discretized form systems of Partial Differential Equations (PDE).
- Models can be implemented accordingly to the object-oriented structure of the code, so that PDEs are expressed in their natural language (equation mimicking):

Transport equation for the scalar field T. Unsteady, convection-diffusion transport equation, ν kinematic viscosity

$$\frac{\partial T}{\partial t} + \nabla \left(\mathbf{U} T \right) - \nabla \cdot \left(\nu \nabla T \right) = 0$$

```
solve
(
    fvm::ddt(T)
  + fvm::div(U, T)
  - fvm::laplacian(mu, T)
    ==
    0
);
```

Momentum equation in conservative from the Navier-Stokes equations

$$\frac{\partial \left(\rho \mathbf{U} \right)}{\partial t} + \nabla \cdot \rho \mathbf{U} \mathbf{U} - \nabla \cdot \nu \nabla \mathbf{U} = -\nabla \mathbf{p}$$

```
solve
(
    fvm::ddt(rho, U)
  + fvm::div(rho, U)
  - fvm::laplacian(mu, U)
    ==
  - fvc::grad(p)
);
```

Correspondence between the implementation and the original equation is clear
Each model answers the interface of its class, but its implementation is separated and independent of the other models
Users have total freedom to create or modify a solver, and can easily reuse functionalities that are pre-compiled into shared libraries
New components do not disturb existing code

Equation mimicking and dictionary

- The specification `fvm` and `fvc` are selected by the user from the `fvSchemes` dictionary in the `system` directory. More details will be reported in next presentation *Cluster configuration and installation* in the Section *OpenFOAM directory organization*.

$$\frac{\partial(\rho \mathbf{U})}{\partial t} + \nabla \cdot \rho \mathbf{U}\mathbf{U} - \nabla \cdot \nu \nabla \mathbf{U} = -\nabla \mathbf{p}$$

```
solve
(
    fvm::ddt(rho, U)
  + fvm::div( rho, U)
  - fvm::laplacian(mu, U)
    ==
  - fvc::grad(p)
);
```

`fvm:laplacian` means an *implict* finite volume discretization for the Laplacian operator, and similarly for `fvm:div` for the divergence operator. The `fvm:ddt` implies an *implicit* time marching scheme. Spatial derivatives are calculated using Gauss' Theorem and temporal derivatives are calculated using Euler-implicit, backward differencing or Crank-Nicolson.
On the other hand the `fvc:grad` means an *explicit* finite volume discretization for the gradient operator. The parenthesis (,) means a product of the enclosed quantities, including tensor product.

Table of Contents

Object Orientation

The use of advanced level C++ as the core programming language brings major benefits to users.
Why C++?

- Advanced error checking at both compile and run time.
- Extremely robust solver and utility executables.
- High speed calculation with efficient memory management and fast linear equation solvers.
- Parallel processing with linear speed up with number of processors (up to thousands).

- This is not a C++ course
- If you interested in C++ programming course, please have a look to our 2014 Course List.

Object Orientation

Basic Components

- Scalars, vectors and tensors with algebra
- Computational mesh, mesh motion, adaptive refinement, topological changes
- Fields (scalar, vector, tensor) and boundary conditions: Dirichelet, Neumenn, acoustically non-reflecting, ...
- Sparse matrix support with linear solver technology. Solve with iterative method the sparse matrix deriving from the FV spatial discretization (Gradient Coniugate, ..)

Discretization Classes

- Implemented as interpolation, differentiation and discretization operators (example...)
- All discretization methods use identical basic components, i. e. common mesh and matrix support. Advantage: better testing and more compact software implementation

Physical Modelling Libraries and Top-Level Solvers

- Libraries encapsulate interchangeable models answering to common interfaces
- Models implement the interface functions, isolated with run-time selection
- Custom-written and optimized top-level solvers for class of physics

Utilities

- Common functionality is needed for most simulation method

Table of Contents

Top-Level Structure: Applications

Assuming your installation has been completed, use the pre-defined alias app to go to the application directory $*FOAM_APP*, where you will find:

`Allwmake`	is used to compile the applications
`bin`	contains the binaries of the applications after compilations
`solvers`	contains the source code of the solvers (alias ⇒ `sol`)
`utilities`	contains the source code of the utilities (alias ⇒ `ut`)
`test`	contains the source code for testing specific features of OpenFOAM

- Applications are divided in two categories: solvers and utilities. OpenFOAM includes, up-to-date, over 80 solver and over 170 utilities.
  - **solvers**, that are each designed to solve a specific continuum mechanics problem
  - **utilities**, that are designed to perform tasks that involve data manipulation (pre- and post-processing tasks, e.g. meshing, data visualisation, code conversion, etc . . .)
- OpenFOAM is distributed with a large number of applications, covering some different aspects of what can be done with OpenFOAM. Don't view these applications as a limit of what you can do, instead view them as *example of what you can do*
- To create a new solver the basic way to do this is to find and copy an application that almost does what is needed, and then to modify it by copy/paste from other applications that has features that are needed.

Top-Level Structure: Applications

- An extensive set of OpenFOAM solvers has evolved (and is growing) that are available to users. OpenFOAM is used mainly for CFD but has found use in other areas such as stress analysis, electromagnetics and finance because it is fundamentally a tool for solving PDEs rather than a CFD package in the traditional sense (i. e. solvers for the NS equations). In $*FOAM_APP*/solvers you find the source code for the solvers arranged according to:

`basic`	`discreteMethods`	`financial`	`lagrangian`
`combustion`	`DNS`	`heatTransfer`	`multiphase`
`compressible`	`electromagnetics`	**`incompressible`**	`stressAnalysis`

- By example, inside $*FOAM_APP*/solvers/**incompressible** you will find the solver source code directories:

`adjointShapeOptimizationFoam`	`boundaryFoam`	`icoFoam`
`nonNewtonianIcoFoam`	`pimpleFoam`	`shallowWaterFoam`
`potentialFreeSurfaceFoam`	`pisoFoam`	`simpleFoam`

- Inside each solver directory you find a *.C file with the same name as the parent directory. This is the main file, where you will find the top-level source code and a short description of the solver.

```
cd icoFoam/
ls
createFields.H  icoFoam.C  icoFoam.dep  Make
```

Table of Contents

OpenFOAM @ CINECA

Experience:

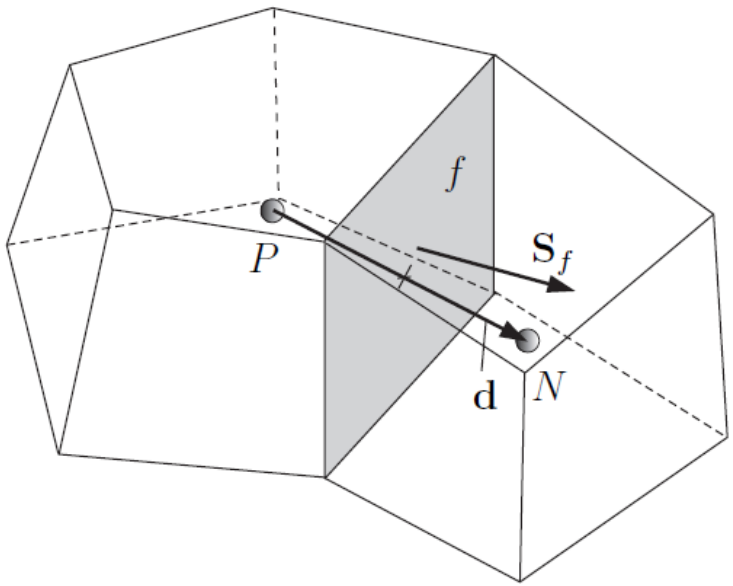
- OpenFOAM installed and tested on our clusters: PLX, EURORA, FERMI
- Analysis of *Current Bottlenecks in the Scalability of OpenFOAM on Massively Parallel Clusters* of OpenFOAM in the framework of PRACE (M. Culpo White Paper)
- Used in 27 Academic project (17 ISCRA + 10 LISA)
- OpenFOAM in FORTISSIMO for the Enabling Manufacturing SMEs to benefit from HPC and Digital Simulation.
  WP417 Cloud-based Computational Fluid Dynamics Simulation in collaboration with Konigsegg, ICON, CINECA and NTUA. The experiments are focused in the use of accurate but very computer intensive transient detached eddy simulation (DES) solvers for Drag and Lift prediction of supercars. More info
- Support for industrial development: 14 projects.

OpenFOAM @ CINECA

TMB project example

Table of Contents

A Glimpse on OpenFOAM Structure



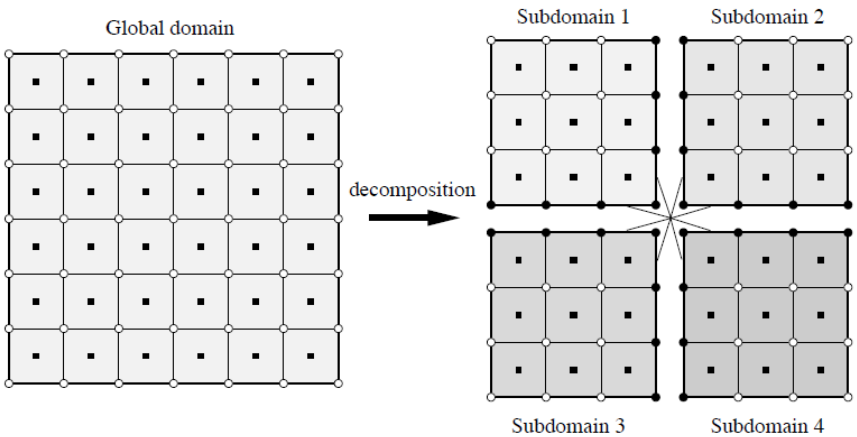
Finite Volume Primer

$$\int_{Cell} \mathrm{div}(A\nabla p)\ dV := \sum_{Faces} \int_f (A\nabla p) \cdot \mathbf{S}_f\ dS$$

Data Representation

- Variables may be associated with:
  1. cells
  2. faces
  3. points
- Time discretized independently

A Glimpse on OpenFOAM Structure



Solution Process: Overview

1. Different solvers, same kernel
   - sparse linear systems
   - iterative Krylov methods
   - SpMV multiplication

Task Decomposition

- Zero-Halo layer approach
- Internal Edges
  1. treated as BCs
- Usual pattern
  1. Interface initialization
  2. Local computation
  3. Interface update
- Local: Diagonal Block
- Interface: Off-Diagonal Blocks

Running in parallel

- The method of parallel computing used by `OpenFOAM` is based on the standard Message Passing Interface (MPI) using the strategy of domain decomposition.
- The geometry and the associated fields are broken into pieces and allocated to separate processors for solution.
- A convenient interface, `Pstream`, is used to plug any Message Passing Interface (MPI) library into OpenFOAM

An analysis has been done in the framework of PRACE 1IP to study

- The current bottlenecks in the scalability of OpenFOAM on Massively parallel clusters.
  - OpenFOAM scales reasonably well up to thousands of tasks.
  - An in-depth profiling identified the calls to the `MPI_AllReduce` function in the linear algebra as core libraries as the main communication bottleneck
  - A sub-optimal performance on-core is due the sparse matrices storage format that does not employ any cache blocking.

M. Culpo, Current Bottlenecks in the Scalability of OpenFOAM on Massively Parallel Clusters, PRACE White Paper, available on-line at www.prace-ri.eu

Missing for a full enabling on Tier-0 Architecture:

- Improve the parallelism paradigm, to be able to scale from the actual $\sim$ 100 procs to the order of $\sim$ 1000 procs.
- improve the I/O, which is a bottleneck for big simulation. For example LES/DNS with hundreds of procs that requires very often saving on disk.

Table of Contents

Remarks: Pro and cons

The open-source nature of OpenFOAM is of fundamental importance but is not the its *only* beneficial feature.
The advantage to using OpenFOAM are:

- Codes are extensible for many customized applications
- the generality of the various OpenFOAM libraries and solvers empowers the user to solve nearly all CFD problems
- the pre- and post-processing interfaces are powerful, and (relatively) user friendly
- dynamic meshes can be used and manipulated for dynamically changing geometry
- object oriented C++ code development strategy makes it convenient for users to incorporate their own models

The barrier of entry into OpenFOAM are also quite non-trivial.
The web-site does not provide a systematic user manual for all codes and applications.
To improve your learning curve on how to use OpenFOAM you have to

- participate in the community organized international workshop
- attending company-offered tutorial courses.
- joining online and/or campus Users Groups or CFD communities
- classroom or personal tutorials by experienced users or instructors.