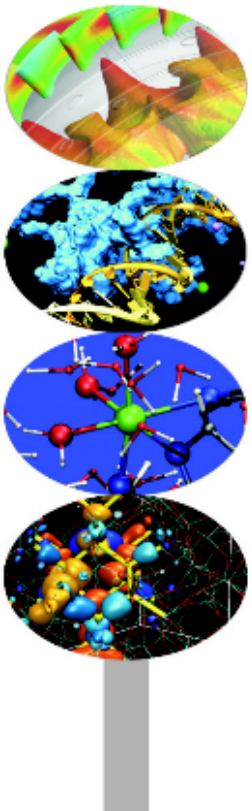
CINECA SCAI
SuperComputing Applications and Innovation

CINECA COURSES High Performance Computing 2014

# HPC enabling of OpenFOAM® for CFD applications



**Towards the exascale: OpenFOAM perspective**

Ivan Spisso

25-27 March 2015, Casalecchio di Reno, BOLOGNA.

SuperComputing Applications and Innovation Department, CINECA

PRACE CINECA

# Actual bottlenecks



Missing for a full enabling on Tier-0 Architecture:

- **Improve the actual parallelism**, to be able to scale from the actual $\sim$ 1,000 procs to at least one order of magnitude ($\sim$ 10,000 or 100,000 procs).
- Improve the I/O, which is a bottleneck for big simulation with big data. For example LES/DNS with hundreds of procs that requires very often saving on disk.

State of the art: A few million cells is now considered relatively small test case. Cases of this size will not scale usefully beyond 1K cores and there is not much to be done to improve this.
Where we are looking at is radical scalability $\Longrightarrow$ The real issues are in the scaling of cases of 100's of millions of cell on 10K+ cores.

# The engine of OF ...



OpenFOAM is first and foremost a *C++ library* used to solve in discretized form systems of Partial Differential Equations (PDE).
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.

# ... and its bottleneck



The scalability of the linear solvers

- The linear algebra core libraries are the main communication bottlenecks for the scalability
- Whole bunch of MPI_Allreduce stems from an algorithmic constraint and is unavoidable, increasing with the number of cores, . . . unless an algorithmic rewrite is proposed.

Generally speaking, the fundamental difficulty is the inability to keep all the processors busy when operating on very coarse grids.

1. Actual bottlenecks
2. The engine of OF and its bottleneck
3. Suggestion / Further Work

# Suggestions



- Choose the linear system solvers: use the geometrical multi-grid solver (GAMG) for very large problems [1]. The GAMG solver can often be the optimal choice, particularly for solving the pressure equation
- Extrapolate the optimal number of cells per cores for your test-case on the selected hardware

[1] W. Briggs, V. Henson, and S. McCormick, *A Multigrid Tutorial: Second Edition* Society for Industrial and Applied Mathematics, 2000.

# (Suggested) Further work



Strategies for improvement the scalability the code

- algorithmic rewrite to reduce the communication: the GAMG solver need to be changed to agglomerate processors as well as cells as the mesh is coarsened to reduce the local transfer costs by matching the communications structure of the machine.
- use the accelerators today available on the HPC cluster (GPU enabling) or Intel-Phi (hybridization with OpenMP)
- co-design, that is design closely integrated to the systemware development: PGAS (Partitioned Global Address Space), UPC (Unified Parallel C), Chapel, X10 and Fortress language developments

All these strategies requires a significant effort and very good OpenFOAM (C++) programmer