CE: AK QA: AK

INTERNATIONAL JOURNAL OF COMPUTATIONAL FLUID DYNAMICS, 2016
VOL. 0, NO. 0, 1–6
http://dx.doi.org/10.1080/10618562.2016.1221503

Taylor & Francis
Taylor & Francis Group

Optimising the Termofluids CFD code for petascale simulations

R. Borrell[a,b], J. Chiva[b], O. Lehmkuhl[b], G. Oyarzun[b], I. RodrÍGuez[b] and A. Oliva[b]

[a]Termo Fluids S.L. C/Magí Colet 8, Sabadell (Barcelona), Spain; [b]Head and Mass Transfer Technological Center (CTTC), Universitat Politécnica de Catalunya, Terrassa (Barcelona), Spain

ABSTRACT

This paper presents some recent efforts carried out on the expansion of the scalability of TermoFluids multi-physics CFD code, aiming to achieve petascale capacity for a single simulation. We describe different aspects that we have improved in our code in order to efficiently run it on 131,072 CPU-cores. This work has been developed using the BlueGene/Q Mira supercomputer of the Argonne Leadership Computing Facility, where we have obtained feedback at the targeted scale. In summary, this is a practical paper showing our experience at reaching the petascale paradigm for a single simulation with TermoFluids.

ARTICLE HISTORY
Received 28 July 2016
Accepted 2 August 2016

KEYWORDS
HPC; parallelisation; scalability; petascale simulations; CFD code; check-pointing

1. Introduction

Since about 10 years, when the clock speed of CPUs stalled due to physical constraints, improvements in computing power of supercomputers have been based on increasing the level of concurrency, i.e. multiplying the number of cores engaged on job executions. Nowadays, we are in a technology disruptive moment with the objective of reaching the exascale paradigm (10^{18} floating point operations per second) with affordable power consumptions. This challenge has driven the hybridisation of the computing systems with the introduction of massively parallel accelerators, which are increasingly tightly coupled with host CPUs at nodes and provide a great concentrated computing power. The hybrid model has been explored for the CFD kernels of TermoFluids (TF) (Oyarzun et al., 2014), but are not the focus of this paper. Here, we focus in the first level of parallelisation, the inter-node connection based on MPI, which remains essential and also requires much higher figures to reach the exascale paradigm. Note that while the intra-node performance aspects, that may bring the most disruptive changes, can be deeply studied even in a single node or on a few of them, deepening on the inter-node parallelisation aspects requires access to supercomputers at the targeted scale. It is also worth noting that despite supercomputers have reached the petascale level since 2008, and the focus is now on the exascale realm (which may be reached in the next decade), nowadays the largest supercomputer (Tianhe-2 from China's National University of Defense Technology) has a peak performance of 33.86 petaflop/s. Moreover, beyond scalability tests, petascale production simulations (i.e. complete simulations engaging a piece of hardware that delivers at least one petaflop/s) are quite rare. Considering that supercomputers are generally shared by many users, we may expect that petascale simulations will be more frequent on leading edge systems when those will be closer to 100 petaflop/s, i.e. on the next generation of pre-exascale systems.

The numerical experiments carried out for the present study were performed on the Mira supercomputer of the Argonne Leadership Computing Facility (ALCF), this is a BlueGene/Q supercomputer which provides a peak performance of 10.07 petaflop/s at running LINPACK benchmark, and is ranked 5th in the current Top500 list (list of June 2015). Mira supercomputer gathers 786,432 CPU-cores by connecting 16-core PowerPC CPUs. Therefore, this is an ideal platform to test and further develop the MPI scalability of our code.

TF is a general purpose multi-physics CFD code based on symmetry preserving finite volume discretisations on unstructured meshes (Lehmkuhl et al., 2007). The turbulence modelisation is based on LES and regularisation models (Lehmkuhl et al., 2012), and the expansion to multi-physics includes, among other phenomena, radiation, combustion, particles, multi-fluid flows or fluid structure interactions (Colomer et al., 2013; Jofre et al., 2015). In terms of parallelism, the largest production simulations performed with TF (engaging up to 5120 CPU-cores) have been simulations of flows with one periodic direction, such as the simulation of bluff bodies (Lehmkuhl et al., 2012) or NACA profiles (Rodríguez et al., 2013), for which a specific direct Poisson solver

CONTACT R. Borrell ricardb@cttc.upc.edu

was developed, showing good parallel efficiency up to 10^4 CPU-cores (Borrell et al., 2011). The objective of the work presented in this paper is jumping an order of magnitude on the MPI-scalability of our general purpose code. Our goal is to prepare the code to run at such level of parallelisation, thus we have included in our study important aspects such as the pre-processing or checkpointing stages. The description of these issues and the corresponding numerical experiments attesting the performance of the new implementations is the main contribution of this paper.

The rest of the paper is arranged as follows. In Section 2, the discretisation method implemented in TF for the Navier Stokes equations is briefly presented. Computational aspects are discussed in Section 3. In Section 4 are presented the numerical experiments performed on Mira supercomputer. Finally, relevant results are summarised and conclusions are given in Section 5.

2. Numerical methods

TFincludes different physical phenomena such as radiation, particles, fluid-structure interactions or interfacial flows. However, in this paper we have focused on the flow solver, which is the core of any simulation performed with TF. The principal set of equations for the simulation of turbulent incompressible flows of Newtonian fluids are the Navier–Stokes (NS) and continuity equations. In an operator-based formulation, the finite volume spatial discretisation of these equations reads

$$\Omega \frac{du_h}{dt} + C\left(u_h\right) u_h + Du_h + \Omega G p_h = 0_h, \quad (1)$$

$$Mu_h = 0_h, \quad (2)$$

where u_h and p_h are the velocity and pressure fields defined at the nodes of the mesh, Ω is a diagonal matrix with the size of the control volumes, $C(u_h)$ and D are the convective and diffusive operators and, finally, M and G are the divergence and gradient operators, respectively. TF is based on a 'symmetry-preserving'/'energy conserving' discretisation. Namely, the convective operator is skew-symmetric ($C(u_h) + C(u_h)^* = 0$, where $C(u_h)^*$ refers to the adjoint of the convective operator), the diffusive operator is symmetric positive-definite and the integral of the gradient operator is minus the adjoint of the divergence operator ($\Omega G = -M^*$). Preserving the symmetries of the continuous differential operators has shown to be a very suitable approach for time-accurate simulations (Lehmkuhl et al., 2012; Rodríguez et al., 2013).

For the temporal discretisation, a second-order explicit one-leg scheme is used. Then, assuming $\Omega G = -M^*$, the resulting fully-discretised problem reads

$$\Omega \frac{u_h^{n+1} - u_h^n}{\delta t} = R\left(\frac{3}{2}u_h^n - \frac{1}{2}u_h^{n-1}\right) + M^* p_h^{n+1}, \quad (3)$$

$$Mu_h^{n+1} = 0_h, \quad (4)$$

where $R(u_h) = -C(u_h)u_h - Du_h$. The pressure-velocity coupling is solved by means of a classical fractional step projection method (Yanenko et al., 1971). In short, reordering Equation (3), an expression for u_h^{n+1} is obtained,

$$u_h^{n+1} = u_h^n + \delta t \Omega^{-1}\left(R\left(\frac{3}{2}u_h^n - \frac{1}{2}u_h^{n-1}\right) + M^* p_h^{n+1}\right), \quad (5)$$

then, substituting this into (4) leads to a Poisson equation for p_h^{n+1},

$$-M\Omega^{-1}M^* p_h^{n+1} = M\left(\frac{u_h^n}{\delta t} + \Omega^{-1}R\left(\frac{3}{2}u_h^n - \frac{1}{2}u_h^{n-1}\right)\right), \quad (6)$$

that must be solved once per time-step.

3. Computing approach

Exploiting the potential of any supercomputer depends on two factors: first, on the sequential performance of the code under consideration, i.e. the performance that can be obtained separately from the different computing units composing the supercomputer; second, on the parallel performance of the code, i.e. on the performance of the parallel implementation that includes inter-CPU data communications and synchronisation points.

In our application context, the first issue is mainly limited by the low arithmetic intensity of the kernels composing our implementation. Considering, for instance, the BlueGene/Q system, it has a peak performance per node of 204.8 Gflop/s and a bandwidth per node of 42.6 Gbytes/s. This means that in order to keep the CPUs busy all the time, for each double precision variable fetched to the cache, 38,5 floating point operations should be performed. Considering, for example, the sparse matrix vector product (SpMV) for a Laplacian matrix discretised over an unstructured mesh, the arithmetic intensity achieved is about 1 flop/double: for each matrix coefficient and its corresponding component of the multiplying vector, a product and a summation are performed. Consequently, no more than 3% of the potential performance of the Mira nodes can be achieved. The same situation repeats on the other operations of the code. This is an underlying limitation of CFD and many other

scientific applications at exploiting the performance of current HPC systems, which are evaluated and ranked by the LINPACK benchmark, which is a benchmark based on dense linear solvers, and as such deals with a completely different computing pattern. In any case, being the CFD a clearly memory bounded application, performance relies on minimising memory transfers and exploiting in the best possible way the intra-node memory hierarchy.

Regarding the parallel performance, the main degradation factors are the inter-process data communications. So, in this paper we have focused on optimising the parts of the code related to MPI communications. Those parts form what can be considered the first level of parallelism, which can be complemented with shared memory parallelism and vectorisation within nodes. This first parallelisation level is based on a geometric domain decomposition. Two types of communications are used in our code: (i) the global reduction operations used in norms, dot products and to evaluate global measures such as the time-step length; (ii) the point-to-point communications required for the halo updates, i.e. for transferring information required to solve the dependencies between unknowns belonging to different subdomains. For the communications of the first type, we did not introduce any change on the code with respect to previous versions, the corresponding `MPI` collectives are just called. On the other hand, the halo updates are performed by means of the non-blocking functions `MPI_Irecv` and `MPI_Isend` that avoid unnecessary synchronisation, deferring this synchronisation to a latter call of the function `MPI_Waitall`. Nonetheless, on the halo updates there was a significant design error that became critical when using tens of thousands of processes. In the previous design, for each process we were using a double pointer (i.e. a pointer of pointers that after its allocation becomes an array of arrays) as a buffer to perform communications. The first array was of dimension equal to the total number of MPI threads or processes. Then the components corresponding to processes with whom communications were required were allocated accordingly. Finally, on the communication process there was loop over the buffer and communications were established with the processes corresponding to non-empty buffer components. This strategy has unnecessary memory and computing costs, it has been substituted by a sparse scheme where each process stores only the list of the other processes it needs to communicate with and then the loop and the buffer are dimensioned accordingly.

Another relevant aspect that influences the performance of the time-integration process are the checkpointing IO operations . Since simulations are generally completed by multiple executions, checkpoints are used to restart simulations form the last point, from a specific point of interest, or from the last point preceding a failure. In TF, the IO operations are managed by means of the HDF5 library (The HDF Group, 1997–2015). Achieving performance on the parallel IO operations with HDF5 library relies on taking advantage of collective operations. However, there are many intrinsic hardware constraints such as the bandwidth of the parallel file system that cannot be overcome. In particular, our layout of data on the hierarchical data format of the HDF5 library consists on one collective data-set for each scalar field and a contiguous region within it reserved to each parallel process engaged on the simulation. Our goal regarding the IO operations of the checkpointing process is that those are fast enough and generate an acceptable overhead.

Finally, the last part of the code that has been optimised to reach petascale simulations has been the pre-processing stage. Generally, on the simulation process the pre-processing stage has a residual cost compared with the overall time integration. However, this statement is not exactly true since complex simulations are not performed at once, generally is required an iterative process to find a proper mesh, to tune some simulation parameters, or to implement accurate boundary conditions. Therefore, since the pre-processing stage may be repeated several times until the final simulation runs, it is also important to minimise its cost to avoid a tedious setting up of the simulation. Here, we consider the pre-processing stage as all the operations performed from the initiation of a simulation until the time integration starts. The inputs of the pre-processing are the mesh file (in HDF5 format) and a file describing the domain decomposition, this can be the output of a mesh partitioner such as METIS (Karypis et al., 2009). Then each process reads its corresponding information from the mesh file and: (i) evaluates all the geometric and topological properties of the mesh that will be required for the time integration; (ii) evaluates the topological properties of the domain decomposition in order to set up the communication scheme for the halo updates; and (iii) performs the set-up of the linear solver. In the previous version of our code, there was a sequential mesh partitioning stage that generated a new HDF5 file with a separated data-set for each parallel process. This strategy was inefficient when engaging $O(10^4)$ parallel processes and finally unworkable on the targeted petascale simulation level. In the present implementation, all the phases of the pre-processing stage are performed in parallel.

4. Numerical experiments

In order to test the performance of TF, we have run its CFD solver under the conditions of the driven cavity case,

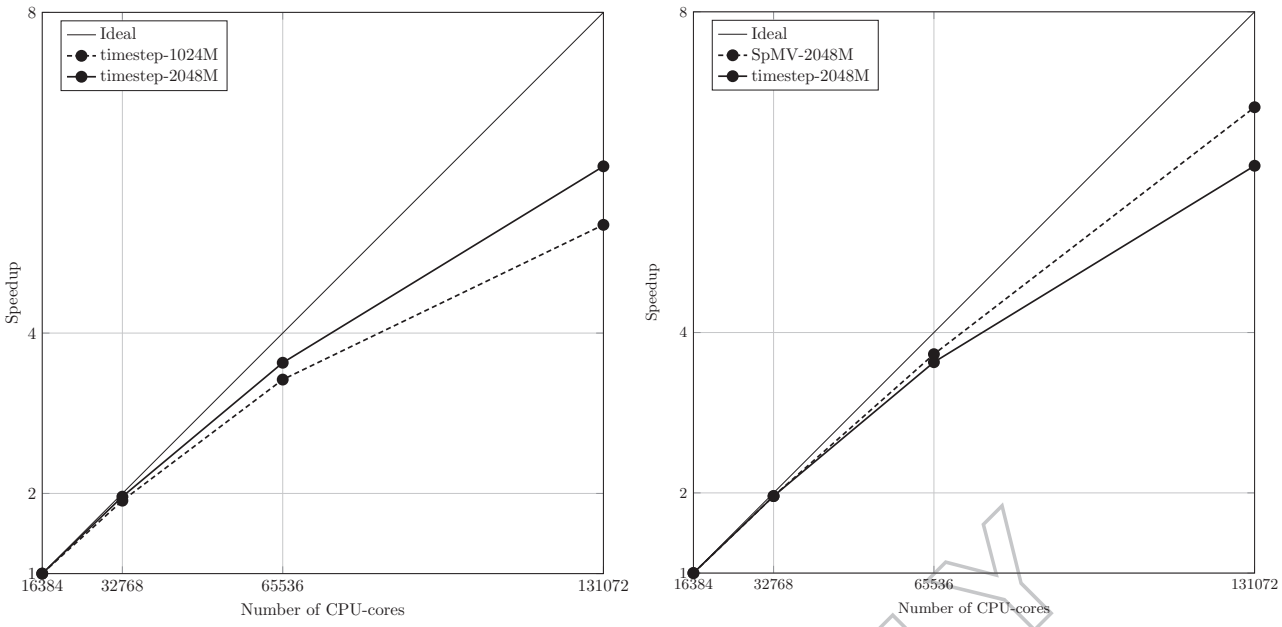

Figure 1. Strong scaling. Left: speedup of the full time-step using two meshes (1024M and 2048M nodes, respectively). Right: speedup of the full time-step and speedup of the SpMV kernel for the largest mesh.

i.e. a box with a horizontal velocity boundary condition at the top. The purpose has not been the completion of any simulation but running along enough time-step iterations to properly measure the average performance of the code per iteration. The meshes have been generated by the extrusion of two-dimensional (2D) unstructured grids, however they are addressed by the code as general three-dimensional (3D) unstructured meshes. The Jacobi preconditioned conjugate gradient method has been used to deal with the Poisson equation.

The first test considered is the strong scaling of the time-integration phase. Results are shown in Figure 1 (left) for two meshes of 1024M and 2048M nodes, generated by the extrusion of an unstructured grid made of one million nodes. The number of CPU-cores ranges from 16,384 up to 131,072, running 16 ranks per node. In both cases, acceleration is observed all the way up to 131,072 cores but, as expected, the larger the mesh size the better the speedup since the relative weight of the communications decreases. In particular, the parallel efficiency achieved is 67% and 76%, respectively. It is important to note that the workload per CPU at the last point, engaging 131,072, is only about 7 and 15K nodes, respectively. In the right part of Figure 1 is compared, for the largest mesh of two billion nodes, the speedup of the time-step and the SpMV kernel for the Laplacian operator. The acceleration of both is almost the same up to 65,536 CPU-cores, what validates the results obtained for the time-step since the SpMV is the dominant kernel. With 131,072 cores, the acceleration achieved with the SpMV is about

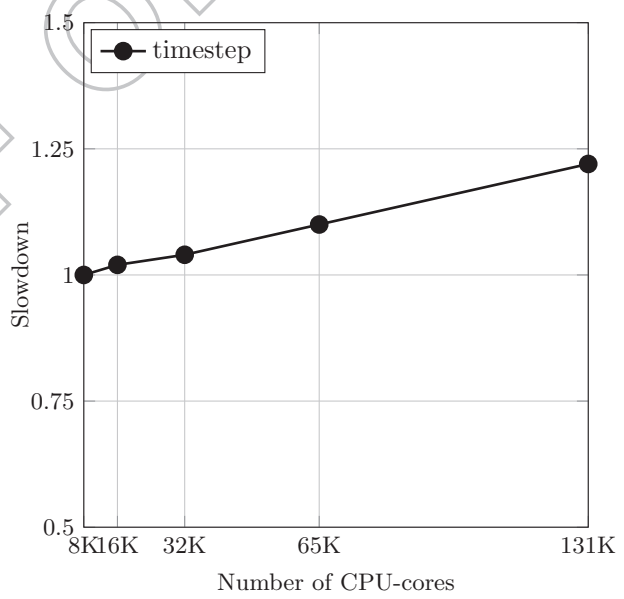

Figure 2. Weak scaling: test for the time-step with a load of 32,250 nodes per MPI task.

10 points above the one for the overall time-step. This can be explained by the collective communications required on the evaluation of norms and other global measures, which end up slowing down the time-step acceleration.

Figure 2 shows a weak scaling test. The load per CPU has been kept constant at the moderate load of 31,250 nodes. The number of CPU-cores is increased from 8192 up to 131,072. It can be observed that while both the size of the problem and the number of CPU-cores are

Table 1. Time(s) spent in the preprocessing and check-pointing phases of TermoFluids, for different number of CPUs and mesh sizes.

CPUs	Mesh size (M)	Pre-process	Check point write	Check point read
8192	256	383	26	17
16,384	512	383	43	24
32,768	1024	386	67	30
65,536	2048	393	106	38

increased by a factor of 16, the cost of the time-step grows only by 22%. Since here we are analysing computing aspects of the code, we have kept the number of iterations of the PCG algorithm constant while increasing the size of the problem. Therefore, this result shows good weak scaling of the kernels involved in the time integration, but does not account for additional iterations required by the linear solver or additional time-steps required during the time integration.

Finally, in Table 1 is shown the time spent in the preprocessing stage and in the check-pointing writing and reading parts, for the same tests run on the previous weak speedup study. Ideally, if the scaling was perfect, the time would remain constant while the number of CPUs and the size of the mesh are proportionally increased. This is almost the situation for the preprocessing stage. On the other hand, as expected, the IO operations through the parallel file system suffer degradation at increasing the number of parallel processes. However, note that in the worst case it takes about 100 s to write a check point for a 2 billion node mesh using 65K CPU-cores. The time-step cost for this case is of 0.8 s, but the checkpointing cost is very acceptable since it is performed between fairly long periods of simulation time.

5. Concluding remark

In this paper, we describe different aspects of TF that have been optimised in order to reach petascale capacity for a single simulation. In particular, we have performed tests engaging up to 131,072 CPU-cores, that sum up a peak performance of about 1.7 peta-flops. The first improvement has been in the inter-CPU communications, in particular in the communication scheme of the halos update process. Notable results have been obtained for both strong and weak scalings of the new version of the code, engaging up to 131,072 CPU-cores. The preprocessing stage has also been optimised avoiding any sequential bottleneck, results show also perfect scalability. Finally, the IO operations have been considered, in this case the scalability is harder due to limitations on the parallel file system, however considering the time required, the overhead generated by the check-pointing within the time-integration phase is almost negligible.

Acknowledgements

This work has been financially supported by the Ministerio de Economía y Competitividad, Spain (ENE2014-60577-R), a Juan de la Cierva posdoctoral grant (IJCI-2014-21034), a PDJ 2014 Grant by AGAUR (Generalitat de Catalunya) and by the CONICYT Becas Chile Doctorado 2012.Calculations have been performed on the Mira supercomputer of the Argonne Leadership Computing Facility. The authors thankfully acknowledge these institutions.

Disclosure statement

No potential conflict of interest was reported by the authors. Q7

Funding

Q8

Ministerio de Economía y Competitividad, Spain [grant number ENE2014-60577-R]; a Juan de la Cierva posdoctoral [grant number IJCI-2014-21034]; a PDJ 2014 Grant by AGAUR (Generalitat de Catalunya) and by the CONICYT Becas Chile Doctorado 2012.

ORCID

A. Oliva http://orcid.org/0000-0002-2805-4794

References

Q9

Borrell, R., O. Lehmkuhl, F.X. Trias, and A. Oliva. 2011. "Parallel Direct Poisson Solver for Discretizations with one Fourier Diagonalizable Direction." *Journal of Computational Physics* 230: 4723–4741.

Colomer, G., R. Borrell, F. Trias, and I. Rodríguez. 2013. "Parallel Algorithms for Transport Sweeps on Unstructured Meshes." *Journal of Computational Physics* 232(1): 118–135.

The HDF Group. 1997–2015 *Hierarchical Data Format*, version 5, http://www.hdfgroup.org/HDF5/.

Jofre, L., R. Borrell, O. Lehmkuhl, and A. Oliva. 2015. "Parallel Load Balancing Strategy for Volume-of-Fluid Methods on 3-D Unstructured Meshes." *Journal of Computational Physics* 282: 269–288. Q10

Karypis, G., and V. Kumar. 2009. *MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System*, Version 4.0, http://www.cs.umn.edu/metis.

Lehmkuhl, O., R. Borrell, I. Rodríguez, C. Pérez-Segarra, and A. Oliva. 2012. "Assessment of the Symmetry-preserving Regularization Model on Complex Flows using Unstructured Grids." *Computers & Fluids* 60: 108–116. Q11

Lehmkuhl, O., C.D. Pérez-Segarra, R. Borrell, M. Soria, and A. Oliva. 2007. "TERMOFLUIDS: A New Parallel Unstructured CFD Code for the Simulation of Turbulent Industrial Problems on Low Cost PC Cluster." *Parallel Computational Fluid Dynamics*, Antalya.

Lehmkuhl, O., I. Rodríguez, R. Borrell, J. Chiva, and A. Oliva. "Unsteady Forces on a Circular Cylinder at Critical Reynolds Numbers." *Physics of Fluids* 26 (12). Q12

Q13

Oyarzun, G., R. Borrell, A. Gorobets, A. Oliva. 2014. "MPI-CUDA Sparse Matrix-vector Multiplication for the Conjugate Gradient Method with an Approximate Inverse Preconditioner." *Computers & Fluids* 92: 244–252.

Rodríguez, I., O. Lehmkuhl, R. Borrell, and A. Oliva. 2013. "Direct Numerical Simulation of a NACA0012 in Full Stall." *International Journal of Heat and Fluid Flow* 43: 194–203.

Yanenko, N.N. 1971. *The Method of Fractional Steps*. Springer-Verlag.