Proceedings of the 11th International Conference on Hydrodynamics (ICHD 2014)
October 19 – 24, 2014
Singapore

PERFORMANCE ANALYSIS OF MASSIVELY-PARALLEL COMPUTATIONAL FLUID DYNAMICS

J. HAWKES
Fluid Structure Interactions (FSI), University of Southampton, UK &
MARIN Academy, Maritime Research Institute Netherlands, Netherlands

S.R.TURNOCK†, S.J. COX*, A.B. PHILLIPS†
Fluid Structure Interactions (FSI) † and Computational Engineering & Design (CED),*
University of Southampton, UK

G. VAZ
R&D Department, Maritime Research Institute Netherlands, Netherlands

As modern supercomputers edge towards exascale, their architectures are becoming more parallel. In order for computational fluid dynamics (CFD) simulations to operate efficiently on newer machines, a complete harmony between hardware, software and numerical algorithms is required. In the work presented here, a typical CFD code is instrumented, and a strong-scalability study performed to identify areas of the execution which require improvement, using the well-known KVLCC2 test case. The effects of changing discretization schemes, mesh structure, turbulence models and linear solvers are all tested. The results show that data-exchange among cores and the inner-loop pre-conditioners both have a large impact on performance in a massively-parallel environment, and should be the focus of future developments.

1. Introduction

The history of the "Top500" supercomputers [1] in figure 1 shows the exponential growth of computing power available to CFD since 1993. The maximum performance, based on a linear algebra tool (LINPACK), has doubled approximately every 14 months, allowing maritime simulations of up to 32-billion cells to be performed [2]. The way in which this growth is achieved varies over time, and programming paradigms for CFD must change as necessary to ensure efficient scaling.

Pre-2004, floating-point operation (FLOP) rates grew exponentially as transistor size decreased. In 2004, predictions on the power consumption of central-processing units (CPUs or simply “chips”) broke down, and since this point the growth of computational performance has been limited by power requirements [3]. CPUs with chip-level multiprocessing (CMP) have become the norm, packaging multiple cores on one chip to improve efficiency. Figure 2 shows this shift in growth mechanism.

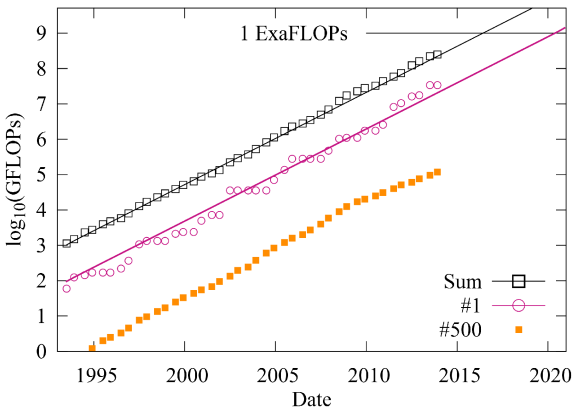

Figure 1: Performance of the Top500 [1] supercomputers. Showing the #1, #500 and sum LINPACK max performance over 20 years, with exponential trend lines.

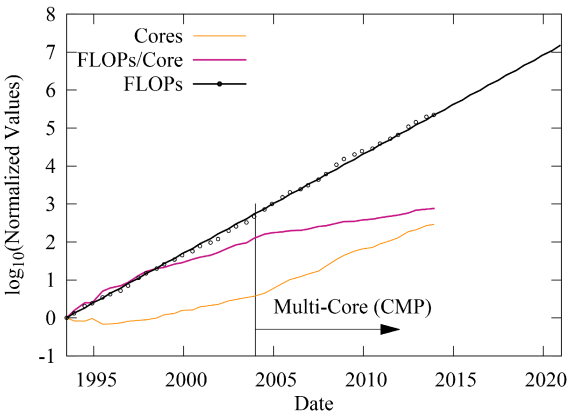

Figure 2: Changes in FLOPs/core, total FLOPs and number of cores, normalized to 1993 values. Values are an average of the Top500 data [1].

CFD has benefited from good "weak scalability" for the last 20 years. As the number of cores has increased CFD problem-sizes have increased too, such that the efficient ratio of cells-per-core has remained roughly constant (anywhere from 20k-100k cells per core, depending on the code, supercomputer and the problem complexity). By current trends, parallelization will be around 14-times greater in 6 years, and one could expect CFD simulations to also be 14-times larger, but there are great challenges in reaching this level.

In 2008, a group of over 20 experts in the supercomputing field were commissioned by DARPA (Defense Advanced Research Projects Agency) to assess the challenges associated with creating the first exascale machine (capable of 1 ExaFLOPs) [4] which should be viable in 2020 according to current growth. Their predictions, which have remained valid over the last 6 years, explain that as power efficiency continues to improve (in terms of pico-Joules per FLOP) the power required to run inter-nodal communication becomes dominant.

Since 2011, there has been a large push towards many-core nodes, which feature O(100) low-power cores to increase efficiency and total FLOP rates using many-core co-processor or graphics-card accelerators [3]. The inter-nodal power bottleneck is likely to drive this trend further and increase intra-node concurrency to O(1k) or O(10k) cores per node [4,5]. With decreasing core clock rates of these many-core nodes, total concurrency on an exascale machine is likely to be 300-times that of the current #1 Top500 machine - Tianhe 2.

Comparing this growth in cores (300x) to the expected growth in CFD complexity (14x), it is clear that there is a need to improve the "strong scalability" of CFD codes. By 2020, it will be necessary to reduce the efficient cells-per-core ratio to approximately 5% of its current value, if predictions hold true.

This paper aims to break down the scalability of a CFD code into its fundamental parts – using the well-known KVLCC2 test case. Profiling tools will be used to extract information on the run-time of the code whilst various user-settings are changed, in an attempt to demystify their effects on scalability. Assessments will be made to determine the areas of the program which exhibit poor strong scalability, as part of ongoing research which aims to develop new techniques to improve the minimum cells-per-core ratio.

2. Experimental Setup

In this section, an overview of the CFD code (ReFRESCO), the supercomputer (IRIDIS4) and the profiling tool (Score-P) is given. The test-case setup is also detailed, and the key scalability studies performed in this paper are explained.

2.1. *ReFRESCO*

ReFRESCO is a viscous-flow CFD code that solves multiphase (unsteady) incompressible flows with the Reynolds-averaged Navier-Stokes (RANS) equations, complemented with turbulence models, cavitation models and volume-fraction transport equations for different phases [6]. The equations are discretized in strong-conservation form using a finite-volume approach with cell-centered collocated variables. The SIMPLE algorithm is used to ensure mass conservation, with pressure-weighted-interpolation (PWI) to tackle the pressure-velocity decoupling issue arising from the collocated arrangement [7].

Time integration is performed implicitly with first or second-order backward schemes. At each implicit time step, the non-linear system for velocity and pressure is linearized with Picard's method - and a segregated or coupled method applied. A segregated approach is adopted for the solution of all other transport equations. All non-linearity is tackled by means of an iterative process so-called the *outer loop*. For each outer-loop iteration, and for each transport equation, an algebraic system of linear equations is solved iteratively until a prescribed residual decay is achieved - this iterative process is the *inner loop*. In order to increase the robustness of the iterative process, implicit (directly on the left-hand-side matrix diagonal) and explicit (directly on the new solution for each variable) relaxation schemes are used.

All numerical schemes used to discretize the transport equations (convection schemes, diffusion, gradients, non-orthogonality corrections, eccentricity corrections) apply their low-order contributions implicitly, to the left-hand side of the equation system; and their higher-order contributions explicitly, to the right-hand side of the system, using values from the previous outer loop.

The implementation is face-based, which permits grids with elements consisting of an arbitrary number of faces and hanging nodes. This also means that all grids, being them structured or unstructured, are treated in the same way inside ReFRESCO, even if usually for unstructured grids more outer-loop iterations are needed to take into account the lower quality metric characteristics.

The code is parallelized using MPI (Message Passing Interface) and sub-domain decomposition. The grids are partitioned in sub-domains, each one having a layer of common cells so-called *ghost-cells*. Each of these sub-domains is calculated in its own MPI process. The ghost-cells are treated as normal cells, as far as the numerical algorithms are concerned, and are therefore handled implicitly.

In many ways, ReFRESCO represents a general-purpose CFD commercial code, with state-of-the-art features such as moving, sliding and deforming grids and automatic grid refinement - but it has been verified, validated and optimized for numerous maritime industry problems.

ReFRESCO is currently being developed at MARIN (Netherlands) [8] in collaboration with IST (Portugal) [9], USP-TPN (University of Sao Paulo, Brazil) [10], TUDelft (Technical University of Delft, the Netherlands) [7], RuG (University of Groningen, the Netherlands) [11] and recently at UoS (University of Southampton, UK).

2.2. *IRIDIS4*

ReFRESCO will be run on the University of Southampton's latest supercomputer, IRIDIS4, which was ranked #179 on the Top500 list of November 2013 [1]. IRIDIS4 has 750 compute nodes, consisting of two Intel Xeon E5-2670 Sandybridge processors (8 cores, 2.6 Ghz), for a total of 12,200 cores and a maximum performance of 227 TFLOPs. Each node is diskless, but is connected to a parallel file system, and has 64GB of memory. The nodes run Red Hat Enterprise Linux (RHEL) version 6.3. Nodes are grouped into sets of ~30, which communicate via 14 Gbit/s Infiniband. Each of these groups is connected to a leaf switch, and inter-switch communication is then via four 10 Gbit/s Infiniband connections to each of the core switches. Management functions are controlled with a GigE network.

2.3. *Profiling Tools*

ReFRESCO will be instrumented with Score-P [12], a profiling tool developed by VI-HPS (Virtual Institute – High Performance Supercomputing). The tool provides compile-time wrappers for ReFRESCO and run-time configuration options which provide useful information on program execution. The results show total call-counts and total time spent within certain functions, and also give insight into MPI communications and load-balancing.

All forms of executable instrumentation are intrusive, since the measuring and buffering of various timers introduces some overhead. Care must be taken to filter the profiling tool, such that only useful information is produced and the overhead is minimized. Profiling tools also disable some compiler optimization (most notably function in-lining), which may have some repercussions.

The total run-time of a profiled ReFRESCO compared to a non-profiled ReFRESCO differs by approximately 2%, and it is assumed that this does not affect the results of the scalability studies.

2.4. *Test Case*

The test case for this investigation will be the KVLCC2 double-body wind tunnel model [13]. The model is simulated at a Reynolds number of 4.6×10^6. The domain and boundary conditions are shown in figures 3 and 4.

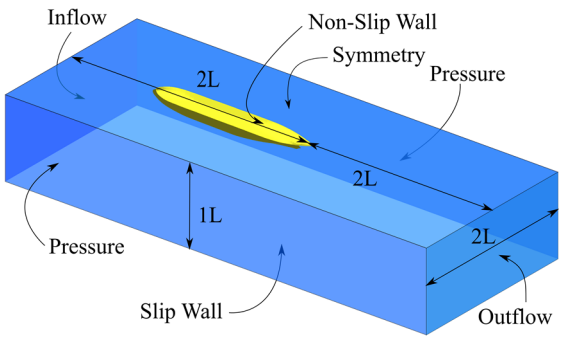

Figure 3: The KVLCC2 domain, not to scale.

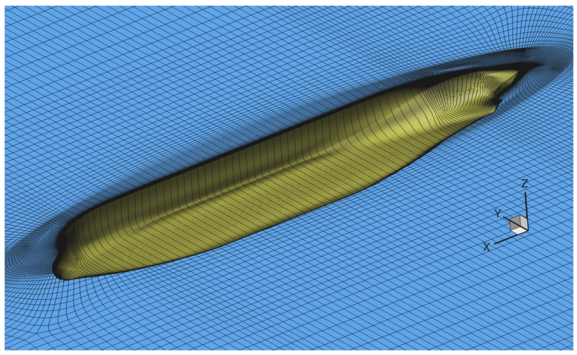

Figure 4: An example of the structured mesh (2.67m cells), from below the waterline.

The following parameters have been used as a baseline for scalability studies, although some variables are tested independently in following studies:

- All tests use a segregated solver.
- Unless specified, a grid of 2.67m cells is used, which covers the current efficient cells-per-core ratio range (20k on ~133 cores, 100k on ~27 cores) with high resolution, and also shows the trends when moving to lower cells-per-core ratio.
- A k-ω, two-equation shear-stress transport turbulence model (SST-2003) is used [14].
- The x-, y- and z-momentum equations, and the turbulence equations, are solved with a Block Jacobi pre-conditioner and a generalized minimal residual (GMRES) solver from the PETSc suite [15]. The inner loop relative ℓ^2-norm convergence tolerance (henceforth ILCT) is provisionally set to 1% (0.01). An explicit outer-

loop relaxation factor of 0.15 is used, as well as an implicit relaxation factor which begins at 0.8 and ramps up to 0.85 over the first 100 iterations.

- The pressure equation is solved with a multi-grid pre-conditioner (ML) and a GMRES solver, using an explicit relaxation factor of 0.1. The ILCT is also chosen as 1%.
- The default convection discretization scheme for the momentum equations is QUICK (Quadratic Upstream Interpolation for Convective Kinematics) with a flux limiter [16], and first-order upwind for the turbulence equations. For all equations, the diffusion terms are discretized using a 2nd-order central scheme. The gradients of any variable (velocity, pressure, turbulence quantities) are discretized using a 2nd-order Gauss-theorem scheme.

2.5. *Scalability Studies*

Initially, a scalability study will be performed using the default settings to provide a benchmark and basic understanding. Following this, various parameters will be changed which represent common user settings, in an attempt to determine their effect, or lack thereof, on strong scalability. The effect of momentum convective flux discretization scheme will be observed, as well as the effects of using different turbulence models, different grids (structured vs. unstructured) and finally, different pre-conditioners and solvers.

3. Scalability Study: A Breakdown of ReFRESCO

In this section, a basic scalability study inheriting all the default settings from section 2.5 is performed over 1000 iterations. Profiling tools are used to break down the execution into distinct sets of subroutines:

- *Assemble* – the assembly of each inner-loop equation system, including construction of the right-hand side vector, for each transport equation.
- *Solve* – the pre-conditioner and solver used to iterate the systems of linear equations.
- *Gradients* – the calculation of gradient values following the iterative solution to each transport equation.
- *Exchange* – data exchange between ghost cells, including transported values, gradients, and other computed values such as eddy viscosity.
- *Other* – the remainder of the above, including initialization routines.

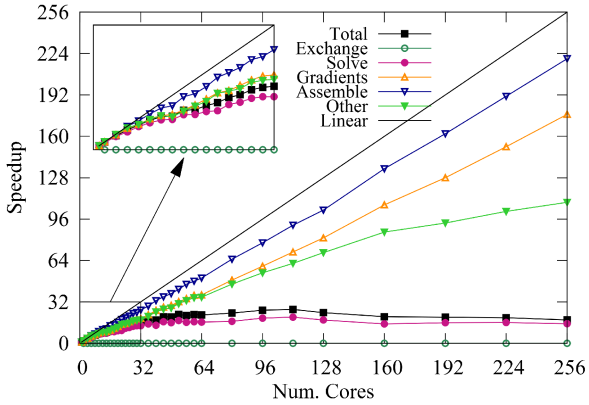

Figure 5: Scalability breakdown of a typical CFD simulation

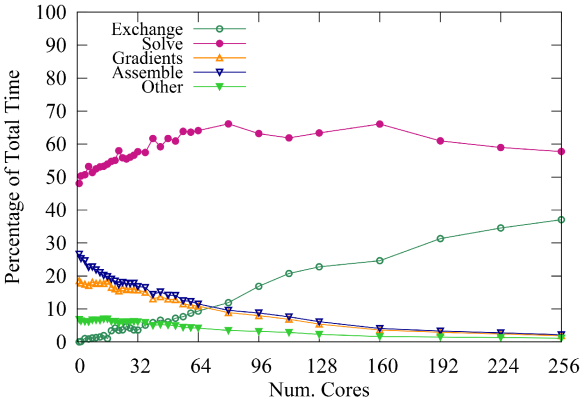

Figure 6: Proportion of total wall-time spent in different subroutines.

Figure 5 shows a typical scalability plot, where the "speedup" is defined as the relative decrease in wall-time with N cores compared to the serial runtime. This speedup can be compared to a linear ("ideal") speedup. Figure 6 shows the proportions of the total wall-time occupied by different sets of subroutines as the number of cores increases.

Figure 5 shows that the *assemble*, *gradients* and *other* parts of the code scale reasonably well, with *other* dropping off at higher core counts due to a number of parallel broadcasts and higher MPI initialization cost. At higher core-counts these parts of the code become a very small proportion of the total run-time due to their efficient scaling, as shown in figure 6. The *exchange* portion of the execution scales inversely with number of cores, which causes a serious bottleneck on strong-scalability. The *solve* routines occupy 48% of the total wall-time in serial operation, and also scale poorly. These are both areas requiring improvement.

4. Scalability Study: Convective Discretization Scheme

The previous scalability study was performed with a QUICK momentum convective discretization scheme with a flux limiter. In the following study, this discretization scheme was varied to ensure that the scalability was independent of the chosen scheme. A first order upwind (UD1), a 50%-central-50%-upwind blend (CD5-UD5), a 90%-central-10%-upwind blend (CD9-UD1), QUICK with limiter and QUICK without limiter (QUICK-NL) were used on a range of structured grids from 317k to 23.4m cells. The solution was allowed to reach an outer loop ℓ^{infty}-norm convergence of 10^{-5}.

The computed form factor [17] is compared to wind-tunnel experimental results [13] in figure 7. The results are presented against the grid coarsening factor, which is the ratio of cells compared to the finest grid.

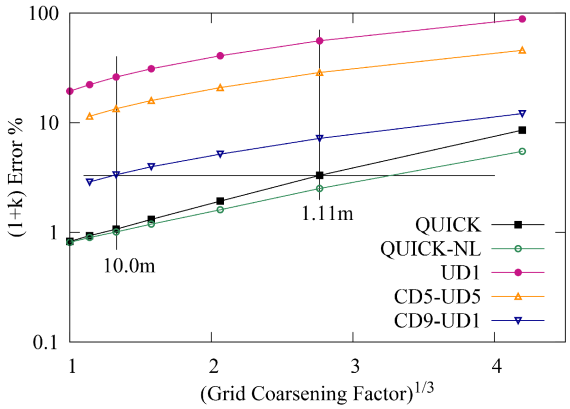

Figure 7: Grid convergence study using various momentum convection discretization schemes, showing form-factor error percentage when compared to experimental results [19].

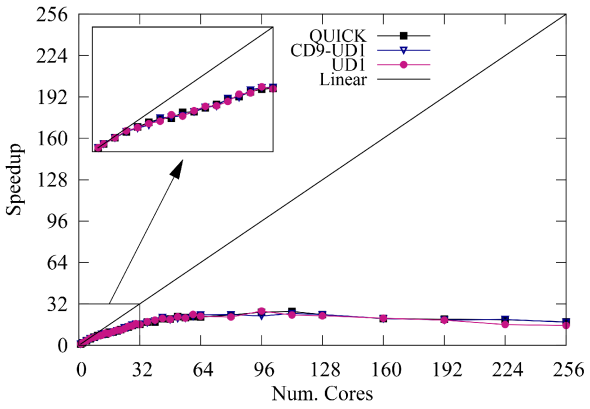

Figure 8: Total speedup for various momentum convection discretization schemes.

As expected, the higher-order schemes provide much higher levels of accuracy, with QUICK matching CD9-UD1 using far fewer cells (1.11m vs. 10.0m). The amount of wall-time required to reach convergence grew exponentially with number of cells, as the number of iterations and the time-per-iteration both increased, but on similar grids all the schemes took similar computational time.

More importantly, the discretization schemes were also compared in a scalability study. The simulations ran for 1000 iterations on a grid of 2.67m cells. The results are shown in figure 8 – no significant changes in scalability were noticed, suggesting that there will be no issues with higher-order schemes in a massively-parallel environment.

Note that ReFRESCO implements a 2nd-order accurate QUICK scheme, since internally the mesh is treated as unstructured – and neighbors-of-neighbors information is not readily available.

5. Scalability Study: Turbulence Models

In this study, the default turbulence model (k-ω SST 2003 [14]) will be compared to a square-root-*kL* (SKL) model [18] and a Spalart-Allmaras (SA) model [19]. Differences are expected due to different amounts of data being computed and exchanged. The simulations are run for 1000 iterations and the speed-up is shown in figure 9.

The SA model required 71% of the total serial run-time of the two-equation models (which were virtually identical in run-time). The SA model has only one transport equation to solve, but also does not incur the overhead of eddy-viscosity and strain-rate computations which, combined, accounts for this wall-time reduction.

Normalized to the serial run-time of each turbulence model respectively, the SA model yields much better scalability. The removal of the second transport equation, eddy viscosity and strain rate computations reduces the amount of data exchange required per outer loop: 2 exchanges versus 8 for the two-equation models (the remaining momentum and pressure equations require a total of 8 exchanges).

6. Scalability Study: Structured vs. Unstructured Grids

Structured grids are often not feasible for practical applications, where the cost of creating the grid is too high. In this study, the scalability of a structured grid simulation is compared to an unstructured one. Unfortunately, it was not possible to create two grids of exactly the same number of cells. An unstructured grid of 12.5m cells was used, and the scalability of two structured grids (10.0m and 15.8m) was linearly interpolated to obtain comparisons. This method is not the most accurate, but any obvious changes in scalability should be clearly visible.

Once again, the simulation was performed to 1000 iterations, but with a more relaxed outer-loop of 0.25 explicit and 0.7 implicit. The results are shown in figure 10.

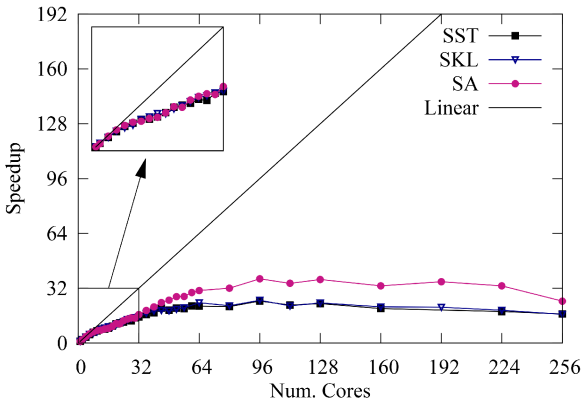

Figure 9: Total speedup for various turbulence models.

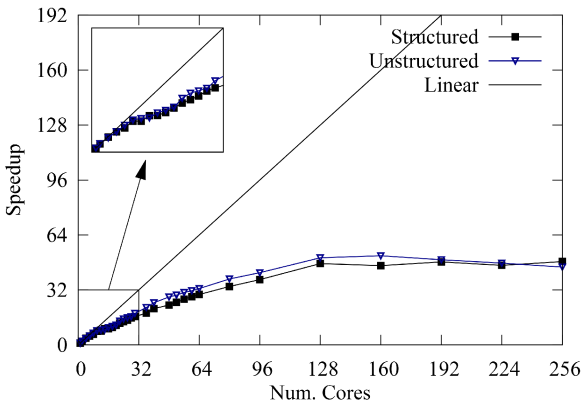

Figure 10: Total speedup with different grids structures.

It is clear that the difference between an unstructured and structured grid is small in terms of scalability. In terms of absolute wall-time, the differences between the two are all less than 5%. This is expected, since ReFRESCO treats the two identically, and the minor differences (in scalability and absolute time) are probably due to the errors associated with linearly interpolating the results.

7. Scalability Study: Pre-conditioners and Solvers

In section 3, the *solve* subroutines were shown to scale poorly, but there are many combinations of pre-conditioner and solver which may perform better. In this scalability study, a number of these combinations will be tested.

The *solve* routines are responsible for iterating the set of linear equations created for each transport equation. The equations are iterated until the solution vector reaches an ℓ^2-norm residual that is a proportion of its initial value. This proportion is the inner loop convergence tolerance (ILCT) – and changing this value greatly affects the load on the solver. It was important to find a reasonable and efficient value for the ILCT before the pre-conditioners and solvers could be tested fairly.

In these tests, starting from a restarted solution of 100 iterations with the default settings, the ILCT is varied between 0.5 (50%) and 10^{-6} (0.0001%). Two outer loop relaxation schemes are tested: implicit 0.9 with explicit 0.1 (tight), and implicit 0.8 with explicit 0.2 (relaxed). In this way, the computational effort associated with linearity and non-linearity of the transport equations can be changed by shifting the focus to the inner loop or outer loop respectively. The total time to reach 10^{-5} ℓ^{infty}-norm outer loop convergence was observed.

Figure 11 shows the results of this study. In general, reducing the inner loop convergence tolerance provides no benefit to overall convergence and serves only to increase the time-per-iteration – except in the case where the linear set of equations is not solved well enough to allow the non-linear iterations to converge. It seems that an ILCT of 0.01 (1%) is a suitable value for numerical stability and efficiency for the outer loop relaxation chosen in section 2.4.

Using this result, the pre-conditioners and solvers can be tested, employing methods from the PETSc suite [15]. For momentum and turbulence, a Block Jacobi, a parallel multi-grid method (BoomerAMG), a parallel additive Schwarz method (ASM), and a parallel ILU pre-conditioner (EUCLID) are tested. For the elliptic pressure equation the same methods are used, but the multi-grid method is replaced with the ML pre-conditioner, which should be faster for elliptic equations.

Two solvers are tested with these combinations: a generalized minimal residual method (GMRES) and a stabilized bi-conjugate gradient squared method (BiCGSTAB).

The scalability results are shown in figure 12, where the key indicates the momentum-turbulence pre-conditioner, the pressure pre-conditioner and the solver for both, respectively. With all pre-conditioners, GMRES and BiCGSTAB perform similarly, so the BCGS results have been removed from figure 12 for clarity. Only the speed-up of the *solve* subroutines are plotted in this graph, having been extracted with the profiling tools. The speed-ups are calculated relative to the serial run-time of BJAC-BJAC-GMRES, in order to make quantitative assessments between total wall-times – the large differences between the pre-conditioners is clearly visible.

Even on low numbers of cores, the Block Jacobi pre-conditioner outperformed all other pre-conditioners, and manages to maintain good speedup to 256 cores. Block Jacobi is a naïve pre-conditioner compared to the others, but the simplicity works in its favour, leading to an overall speedup. EUCLID and BoomerAMG both offered poor scalability, while the additive Schwarz and ML methods offered mediocre scalability. Some of the poorer methods could not complete on a low number of cores, as they exceeded wall-time limitations.

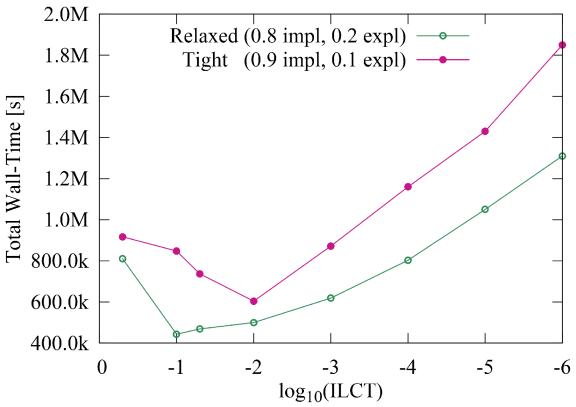

Figure 11: Effects of ILCT on total wall-time using two different outer loop relaxation factors.

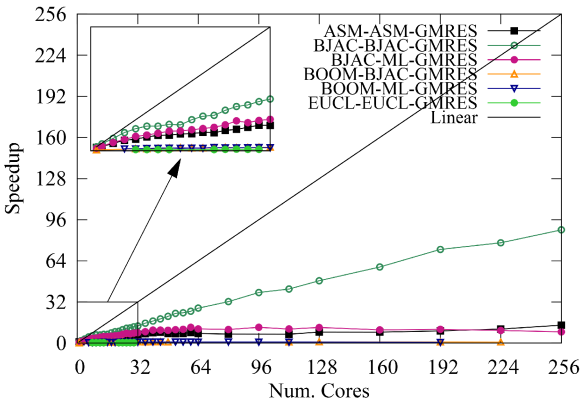

Figure 12: *Solve* scalability with different pre-conditioners and solvers, normalized to BJAC-BJAC-GMRES.

From these studies, it is clear that a Block Jacobi pre-conditioner for all equations is the best starting point for highly scalable CFD. The results of the initial breakdown of ReFRESCO (section 3) are repeated here using these pre-conditioners and the GMRES solver. Figure 13 shows that the *solve* routines have become less of a bottleneck; but in figure 14 it is clear that these routines still take a large part of the execution time. Ignoring the *exchange* routines, which must be improved, *solve* occupies 56% of the remaining execution time on 256 cores.

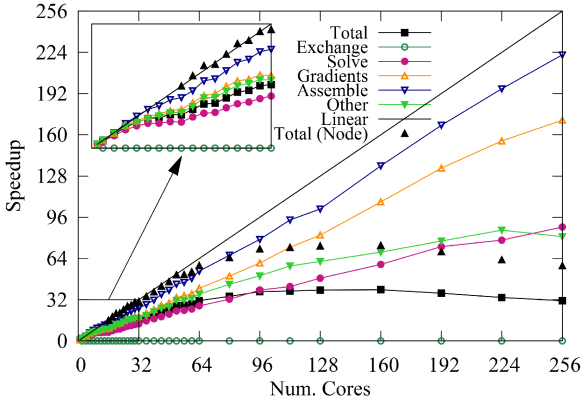

Figure 13: A scalability breakdown of ReFRESCO using more-scalable pre-conditioners (Block Jacobi).

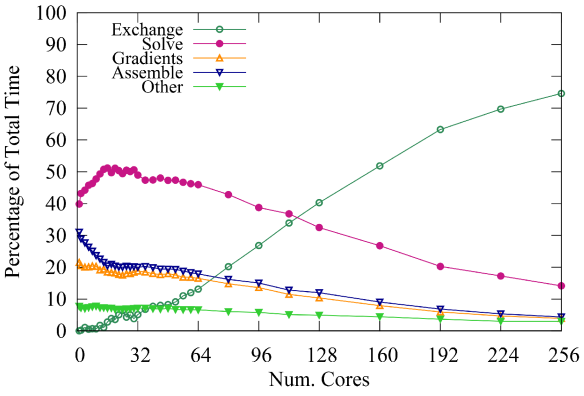

Figure 14: Proportions of total wall-time spent in different routines using more-scalable pre-conditioners (Block Jacobi)

Often, this speed-up is plotted relative to the number of nodes only, such that rather than normalizing to serial run-time, the results are normalized to the run-time on one node (see Figure 13 – "Total (Node)"). This hides the intra-node inefficiency and shows that inter-node scaling is relatively good up to ~42k cells per core (64 cores). For next-generation computing the pressure is on intra-node concurrency, so this view is not the most helpful; however it is important to note that this inter-node scaling compares well with other codes. STAR-CCM+ [20] and ANSYS CFX/FLUENT [21] both show linear speed-up with number of nodes similar to ReFRESCO, but do not present data beyond the efficient range. The ANSYS results also show similar intra-node efficiency (10-12x speedup on 16 cores) compared to ReFRESCO (~10x speedup on 16 cores) – but it is hard to assess fairly without identical hardware and case setup.

8. Conclusions

The breakdown of a typical CFD code was performed in section 3, showing the scalability of the code and the cost of various routines at higher core-counts. It was noted that the *exchange* routines responsible for updating ghost-cell values between cores were costly and non-scalable. There will be a need to explore alternative communication paradigms for massively-parallel CFD – perhaps taking more advantage of shared (intra-node) memory.

The choice of mesh structure and discretization scheme was shown to have very little effect on strong scalability, which is promising for next-generation higher-order CFD simulations. Turbulence models which required more communication caused decreased overall scalability, but improvements in the *exchange* routines may negate this.

Finally, the choice of pre-conditioner had a large effect on the scalability of the code. The more complex pre-conditioners, such as the multi-grid methods, scaled poorly compared to the straight-forward Block Jacobi method. The solver itself seems to have little effect, although only GMRES and BICGStab were tested.

Even the best pre-conditioners caused the *solve* routines to scale poorly compared to the *assembly* and *gradients* routines, and the *solve* routines also take the largest proportion of wall-time. This should be an area for development – especially considering the large differences between the existing pre-conditioners.

Although attempts have been made to generalize the results obtained in these studies, it is important to note that the complex interaction between parts of the code make it difficult to do so. Moreover, only a simple KVLCC2 case has been tested here, and it is assumptive to extrapolate these results to more complex cases – especially where extra features are added, such as cavitation models, free-surfaces or moving interfaces. In addition, some significant parameters have not been tested – such as the use of a coupled solver.

Despite this, there are certain parts of the code such as the *solve* routines and ghost-cell *exchange* routines that can be identified as areas for improvement. These should be tackled in order to allow CFD simulations to continue to benefit from next-generation supercomputing facilities.

Acknowledgments

The authors acknowledge the use of the IRIDIS HPC Facility, and associated support services at the University of Southampton. Approximately 150k core-hours were used for this work over ~700 jobs.

References

1. *Top500*, top500.org (accessed Feb. 2014).
2. T. Nishikawa, Y. Yamade, M. Sakuma, C. Kato, *Fully resolved large eddy simulation as an alternative to towing tank tests – 32 billion cells computation on K computer*, Numerical Towing Tank Symposium (NUTTS), Mülheim, Germany, September 2nd-4th (2013).
3. H. Simon, *Why we need exascale and why we won't get there by 2020*, Optical Interconnects Conference, Santa Fe, New Mexico, August 27th (2013).
4. P.M. Kogge (editor), *Exascale computing study: technology challenges in achieving exascale systems*, Univ. of Notre Dame, CSE Dept. Tech. Report TR-2008-13 (2008).
5. J. Shalf, *The evolution of programming models in response to energy efficiency constraints*, Oklahoma Supercomputing Symposium, Oklahoma, USA, October 2nd (2013).
6. G. Vaz, F. Jaouen and M. Hoekstra, *Free-surface viscous flow computations. Validation of URANS code FRESCO*, In Proceedings of OMAE, Hawaii, USA, May 31st-June 5th (2009).
7. C.M. Klaij and C. Vuik, *Simple-type preconditioners for cell-centered, collocated, finite volume discretization of incompressible Reynolds-averaged Navier-Stokes equations*, International Journal for Numerical Methods in Fluids, 71(7), pp. 830-849 (2013).
8. L. Eca and M. Hoekstra, *Verification and validation for marine applications of CFD*, 29th Symposium on Naval Hydrodynamics (ONR), Gothenburg, Sweden, August 26th-31st (2012).
9. F. Pereira, L. Eca and G. Vaz, *On the order of grid convergence of the hybrid convection scheme for RANS codes*. CMNI, pp. 1–21, Bilbao, Spain, June 25th-28th (2013).
10. G.F. Rosetti, G. Vaz, and A.L.C. Fujarra, *URANS calculations for smooth circular cylinder flow in a wide range of Reynolds numbers: solution verification and validation*, Journal of Fluids Engineering, ASME, July, p. 549 (2012).
11. H. Bandringa, R. Verstappen, F. Wubbs, C. Klaij and A.v.d. Ploeg, *On novel simulation methods for complex flows in maritime applications*, NUTTS, Cortona, Italy, October 7th-9th (2012).
12. VI-HPS, *Score-P*, vi-hps.org/projects/score-p, version 1.2.3, (accessed Nov. 2013).
13. S. Lee, H. Kim, W. Kim and S. Van, *Wind tunnel tests on flow characteristics of the KRISO 3,600 TEU containership and 300K VLCC double-deck ship models*, Journal of Ship Research, 47(1), pp. 24-38 (2003).
14. F.R. Menter, M.Kuntz, and R. Langtry, *Ten years of industrial experience with the SST turbulence model*, Turbulence, Heat and Mass Transfer 4, Antalya, Turkey, October 12th-17th (2003).
15. S. Balay et al., *PETSc user manual*, mcs.anl. gov/petsc, Argonne National Labatory, ANL-95/11 – Rev. 3.4 (2013).
16. M. Hoekstra, *Numerical simulation of ship stern flows with a space-marching Navier-Stokes method*, PhD Thesis, Delft University of Technology (1999).
17. A.F. Molland, S.R. Turnock and D.A. Hudson, *Ship resistance and propulsion: practical estimation of ship propulsive power*, Cambridge, GB, Cambridge University Press, p.70 (2013)
18. F.R. Menter, Y. Egorov and D. Rusch, *Steady and unsteady flow modelling using the $k-\sqrt{kL}$ model*, Turbulence, Heat and Mass Transfer 5, Dubrovnik, Croatia, September 25th-29th (2006).
19. P.R. Spalart and S.R. Allmaras, *A one-equation turbulence model for aerodynamic flows*, AIAA Paper 92-0439 (1992).
20. STAR-CCM+, *Performance benchmark and profiling*, HPC Advisory Council (Best Practices), July (2010).
21. D. Mize, *Scalability of ANSYS applications on multi-core and floating point accelerator processor systems from Hewlett-Packard*, Hewlett-Packard, February 7th (2012).