OpenMP performance with an Euler code on an Intel based personal computer

D. Norrison[1] E. Ly[2]

(Received 30 July 2007; revised 4 May 2008)

Abstract

Personal computers of the past were limited to running computational fluid dynamics codes in serial mode. With the advent of multicore technology and suitable compilers, personal computers nowadays can execute codes in a parallel fashion similar to that of supercomputers and cluster computer systems. This article investigated what performance can be achieved when executing an aerodynamic code on an Intel quad core based personal computer with an OpenMP compiler in a Windows environment. The code solved the Euler equations to find the flowfield around a NACA 0012 aerofoil on an O-type boundary fitted structured grid system. A speedup of up to 350% was obtainable in double precision accuracy. The reduced computation time means that, for small scale problems, more accurate Euler codes can replace commonly used transonic small disturbance codes. For larger problems, this information serves as a reference for developers of hybrid MPI/OpenMP algorithms for cluster computer systems.

See http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/345 for this article, © Austral. Mathematical Soc. 2008. Published June 5, 2008. ISSN 1446-8735

Contents

1 Introduction

Inadequate and affordable computer power has plagued computational fluid dynamics (CFD) research since its inception. Consequently, the claim made by Harvard Lomax [18] in the late 1960s that CFD would replace wind tunnel experiments has not yet materialised. Instead, problems that required investigation were simplified, so that solutions were obtained within an acceptable turnaround time using the computing platforms available. Often the only way to obtain results in a timely manner, for demanding high level aerodynamics solvers based on Euler or Navier–Stokes equations, required employing a very expensive supercomputer or cluster computer system. Consequently, research predominantly focuses on large-scale computations and associated issues using such systems [3, 4, 8]. However, recent developments offer a tantalising insight into new computation accelerating technologies for ordinary desktop personal computers (PCs). These include using relatively inexpensive high end graphics card based solutions with AMD/ATI's Close

To the Metal (CTM) [1] and Nvidia's Compute Unified Device Architecture (better known as CUDA) [16], which offer performance of up to 500 Gflops per card for single precision calculations (double precision versions have just been released). An alternative is Intel's Tera-scale research project [7], which created an 80 core prototype processor that delivers in excess of 1 Tflops, although currently the core count is limited to four for most commercially available general purpose processors.

Here we ask what sort of performance can a regular PC offer for less demanding CFD applications, particularly given that multiple processing cores can now be accommodated in a single socket rather than resorting to the multi-socket architecture of the past? A finite difference based scheme, implementing Van Leer flux blending [21], was developed to solve the two dimensional Euler equations, in double precision accuracy, on an O-type structured grid system [14, 15]. The grid system was generated by another code, implementing an efficient method of false transients, coupled with an approximate factorisation (AF) technique [12, 13] and a variable time step cycling process [11]. The computations for flowfields around a NACA 0012 aerofoil at 2° angle of attack were performed on a PC with an Intel 2.4 GHz quad core processor using an OpenMP compiler in a Windows environment.

2 Grid generation

The first step in solving the Euler equations involves establishing a smooth boundary fitted grid system around the aerofoil. To do this the cartesian coordinates in the physical domain, $\boldsymbol{r} = (x, z)$, are mapped to general curvilinear coordinates in the computational domain, $\boldsymbol{\vartheta} = (\xi, \zeta)$, via a relation $\boldsymbol{\vartheta} = \boldsymbol{\vartheta}(\boldsymbol{r})$. The mapping is one to one to ensure grid lines of the same family do not cross each other, and provides a smooth grid distribution with minimum skewness [6]. The mapping is constructed by specifying the desired grid points on the boundary with the interior point distribution determined

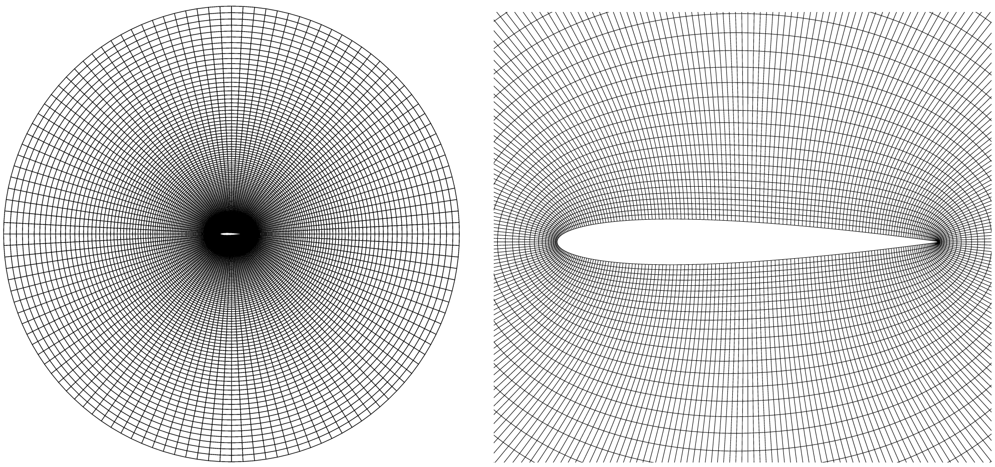
Figure 1: Grid system used in the simulations: (left) full view; (right) grid region near the NACA 0012 aerofoil.

through the solution of a system of Poisson's equations. Additional grid clustering in a specific region of the physical domain can be enforced via the source terms of the equations. The grid equations are solved by the method of false transients, coupled with an AF technique [12, 13] and a variable time step cycling process [11] to further enhance the convergence rate of the grid generation process. The process is efficient and robust, and is well documented [14, 15]. Figure 1 shows the grid system used for the simulation, which has the farfield boundary located at a distance of eight chord lengths from the centre of the aerofoil. To investigate the effect grid size has on performance, coarse (100 streamwise points by 50 radial points), medium (200 by 100) and fine (400 by 200) meshes were created. A distribution ratio of two to one was chosen, so that the cells would have an aspect ratio of one to one near the aerofoil to minimise grid line skewness.

3 Euler solver with flux blending technique

The adiabatic flowfield around the aerofoil, without body forces, is assumed to be governed by the unsteady Euler equations, written in strong conservation law form with flux variables as the dependent variables [5, 6]. The equations were nondimensionalised with freestream fluid density, freestream fluid speed and aerofoil chord length, resulting in

$$\widehat{\boldsymbol{Q}}_\tau + \widehat{\boldsymbol{E}}_\xi + \widehat{\boldsymbol{G}}_\zeta = \boldsymbol{0}\,, \tag{1}$$

where

$$\widehat{\boldsymbol{Q}} = \tfrac{1}{J}\boldsymbol{Q}\,, \quad \widehat{\boldsymbol{E}} = \tfrac{1}{J}\big(\boldsymbol{E}\xi_x + \boldsymbol{G}\xi_z\big)\,, \quad \widehat{\boldsymbol{G}} = \tfrac{1}{J}\big(\boldsymbol{E}\zeta_x + \boldsymbol{G}\zeta_z\big)\,, \tag{2}$$

and J is the Jacobian of transformation. In the above and subsequent equations, bold face letters represent vectors, and subscripts represent differentiation as in $\boldsymbol{Q}_\tau = \partial\boldsymbol{Q}/\partial\tau$. The solution vector, $\boldsymbol{Q}$, and flux vectors, $\boldsymbol{E}$ and $\boldsymbol{G}$, in Equation (2) are defined by

$$\boldsymbol{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho w \\ \rho e_t \end{bmatrix}, \quad \boldsymbol{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u w \\ u(\rho e_t + p) \end{bmatrix}, \quad \boldsymbol{G} = \begin{bmatrix} \rho w \\ \rho w u \\ \rho w^2 + p \\ w(\rho e_t + p) \end{bmatrix}, \tag{3}$$

where ρ is the fluid density, (u, w) are the cartesian fluid velocity components, e_t is the total energy per unit mass, and p is the fluid pressure determined from

$$p = (\gamma - 1)\left[\rho e_t - \tfrac{1}{2}\rho\big(u^2 + w^2\big)\right]. \tag{4}$$

Here γ is the ratio of specific heats, which is about 1.4 for ambient air.

The flux vectors in Equation (1) are split according to $\widehat{\boldsymbol{E}} = \widehat{\boldsymbol{E}}^+ + \widehat{\boldsymbol{E}}^-$ and $\widehat{\boldsymbol{G}} = \widehat{\boldsymbol{G}}^+ + \widehat{\boldsymbol{G}}^-$, resulting in

$$\widehat{\boldsymbol{Q}}_\tau + \left(\widehat{\boldsymbol{E}}^+_\xi + \widehat{\boldsymbol{E}}^-_\xi\right) + \left(\widehat{\boldsymbol{G}}^+_\zeta + \widehat{\boldsymbol{G}}^-_\zeta\right) = \boldsymbol{0}\,. \tag{5}$$

In the finite difference scheme, forward flux terms (plus superscript) are discretised using a backward difference rule, and a forward difference rule for backward flux terms (minus superscript). When compared to centred schemes this brings some advantages, such as superior dissipation and dispersive properties, and up to twice the stability bound [19]. Van Leer flux blending [21] is used to solve the equation since it overcomes the spurious oscillations (that appear at sonic transitions and stagnation points) produced by the Steger and Warming flux splitting [9, 19] technique.

Since the governing equation system is hyperbolic in time, for steady flow simulations the solution process is marched in time until a steady state solution is obtained. For simplicity and since explicit schemes are well suited to parallel execution, a second order accurate (in both space and time) flux splitting version of MacCormack's scheme [5] is employed. MacCormack's scheme involves a predictor-corrector sequence at each time level. In the predictor step, the time derivative is approximated by a first order forward time difference rule while all spatial derivatives are approximated by first order backward and forward difference rules as appropriate, yielding

$$\begin{aligned}\widehat{\boldsymbol{Q}}^{n+1}_{i,k} = \widehat{\boldsymbol{Q}}^{n}_{i,k} - \frac{\Delta\tau}{\Delta\xi}\left[\widehat{\boldsymbol{E}}^{+}_{i,k} - \widehat{\boldsymbol{E}}^{+}_{i-1,k} + \widehat{\boldsymbol{E}}^{-}_{i+1,k} - \widehat{\boldsymbol{E}}^{-}_{i,k}\right]^{n} \\ - \frac{\Delta\tau}{\Delta\zeta}\left[\widehat{\boldsymbol{G}}^{+}_{i,k} - \widehat{\boldsymbol{G}}^{+}_{i,k-1} + \widehat{\boldsymbol{G}}^{-}_{i,k+1} - \widehat{\boldsymbol{G}}^{-}_{i,k}\right]^{n},\end{aligned} \tag{6}$$

where $\Delta\tau$ is the time step, $\Delta\xi$ and $\Delta\zeta$ are the grid spacings in the streamwise and radial directions. Since the grid spacings can be selected arbitrarily, they are set to unity. This simplifies the related expressions that are to be evaluated by the scheme, and hence helps to reduce the required computation time and errors resulting from evaluating the expressions numerically. The corrector step is more involved, and was presented by Steger and Warming [19]. The MacCormack scheme has been shown to be conditionally stable [5, 19] provided that the time step is not too large. A time step of about 4.3×10^{-4} (equal to two microseconds), found empirically, seemed to provide a numerically stable solution with the boundary conditions used.

It is essential that proper boundary conditions are implemented in the numerical solution procedure, otherwise the problem will be ill posed and detrimentally affect the computations. The slip condition is enforced to ensure a nonzero velocity exists at a tangent to the aerofoil surface for inviscid flow. As there is no mass flow into or out of the aerofoil, the relationship $w = -u\zeta_x/\zeta_z$ holds at the aerofoil surface. Numerical experiments have shown that it is not necessary to enforce the Kutta boundary condition in the wake region [5]. The artificial branch cut made in the grid system has periodic boundary conditions and was treated in the manner suggested by Thompson, Warsi and Mastin [20]. Dirichlet boundary conditions were applied to the farfield boundary due to their simplicity. Although being overprescribed, there is little difference in lift forces between this and well-posed characteristic boundary conditions [2] which are harder to implement. Freestream fluid values at sea level conditions for ambient air were used as initial conditions. The solution was deemed converged when the Euclidean norm of the difference of the computed solution between any two consecutive time levels, divided by the total number of grid points, was less than 10^{-10} for each of the flux variables, or when the scheme reached 5×10^5 iterations.

4 Computing resources

In terms of software, for parallel programming mainly two options exist, notably Open Multi-Processing (OpenMP) and message passing solutions such as the Message Passing Interface (MPI). The latter is used with cluster computer systems, where data is transferred between nodes. While it can also be used for multiple processors on a single node, the programming effort required is much higher than for OpenMP which is designed to share memory on an individual node. For this reason, OpenMP was used here.

In the code, the compiler directive `!$OMP PARALLEL DEFAULT(PRIVATE) SHARED(Qhat,...)` marked the start of the parallel section inside the iter-

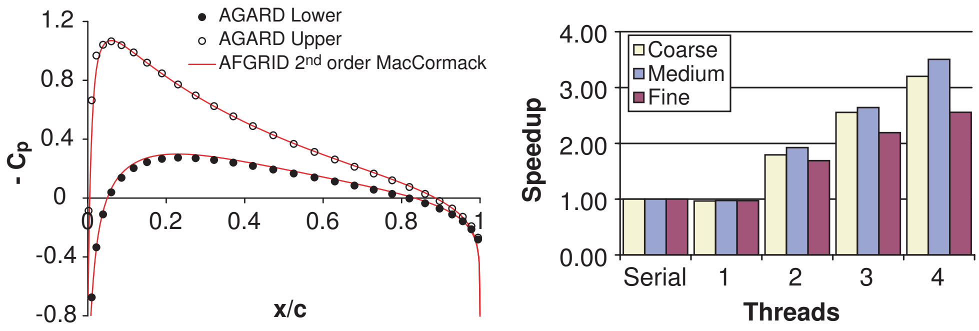

Figure 2: (left) Pressure coefficient distribution for NACA 0012 aerofoil; (right) comparison of computing speedups for different thread modes.

ation loop. The thread numbers were limited with `!$OMP NUM_THREADS()`. The directives `!$OMP DO` and `!$OMP END DO` were placed at the start and end of each of the two loop blocks (predictor and corrector steps), where intensive computations took place to calculate the flow variables for each grid point at each time level. The parallel section was then terminated after the second loop with the `!$OMP END PARALLEL` directive.

The Intel Visual FORTRAN Compiler 9.1 (standard edition) for Windows, which supports OpenMP, coupled with Microsoft Visual Studio 2005, was expected to offer the best performance for Intel processor under investigation. The compiler options were set to optimise for maximum speed on Pentium IV and additional Intel processors, together with `/QaxB` and `/fast` which were empirically found to reduce execution time. The personal computer consisted of an Intel 2.4 GHz Core 2 Quad Q6600 processor with 8 Mb L2 cache and 1,066 MHz front side bus, Gigabyte motherboard with an Intel G965 chipset and Corsair 2x1 Gb DDR2 800 MHz dual channel memory, running Microsoft Windows XP Professional Service Pack 2.

5 Results and discussions

For investigation purposes, flowfields around a NACA 0012 aerofoil at 2° angle of attack, as depicted in Figure 1, were computed on an O-type structured grid with coarse, medium and fine meshes. The freestream Mach number is 0.63 and the Reynold's number is 1.467×10^7. The calculated pressure distribution is presented with the well known numerical inviscid solution by Lock [10] in Figure 2.

Adopting the serial code computation time as the benchmark, the performance results illustrated in Figure 2 show that a substantial reduction in execution time is achieved with a multicore PC. However, with only one thread the speedup is reduced by 2.5% to 3.3% due to the overhead associated with enabling OpenMP. As the number of threads increased, the speedup associated with the quad core processor varied approximately linearly with the thread number. Furthermore, a larger speedup is seen with the coarse and medium grids when compared to the fine grid. In regards to the coarse and medium grid results, a maximum speedup of 350% (which is slightly below the ideal value of 400%, most likely due to communication overhead) was achieved using four threads. The disparity between the two smaller grids is attributed to the OpenMP overhead to computation ratio per iteration being higher for the coarse grid than for the medium grid. In other words, each outer loop iteration performs computations for all of the grid points in the streamwise direction along a given radial level. Since the medium grid contains more nodes than the coarse grid, the computation work done per iteration is greater for the medium grid while the OpenMP overhead is the same for each grids. Therefore, the proportion of time per iteration spent by OpenMP to distribute the workload is larger for the coarse grid compared to the medium grid, and this results in a slight reduction in the speedup for the coarse grid. As for the fine grid, the speedup rose to a maximum of 256%. This significant reduction in performance, compared to the coarser grids, is suspected to be caused by inadequate front side bus bandwidth. The coarse and medium grids occupied approximately 1.5 and 6.1 Mb of memory, respectively, which the faster cache memory could accommodate. Thus, the computations were limited by processor speed, whereas the fine grid required about 24 Mb, which could only fit in the slower main memory whose accessibility restricted the computation speed.

The results for the smaller grids have demonstrated that the new multicore architecture offers similar scaling performance to older shared memory multiprocessor architecture [8]. The execution time using four threads was under ten minutes for the coarse and medium grids, and about fifty minutes for the fine grid. When moving from two to three dimensional simulations, computer requirements increase rapidly both in terms of processing power and memory. While memory is typically limited to a maximum of 8 Gb for a PC at present, which would be sufficient for three dimensional simulations, the computation time for a fine grid is expected to be in the order of days. For these types of larger simulations, a cluster computer system could produce results more rapidly although this would be dependent upon factors including the communication to computation ratio (which may limit scalability) and accessibility to the cluster (without waiting in a queue).

As computational power increases, more versatile and general but less computationally efficient codes will be adopted [17]. As an example, with the processing speed of a quad core PC, a two dimensional Euler solver could replace popular full potential and transonic small disturbance [12, 13] solvers which are limited to the potential flow regime, supercritical flows with weak embedded shock waves, thin aerofoils and small angle of attacks.

6 Concluding remarks

The scaling performance of an Intel quad core processor using OpenMP to accelerate a two dimensional CFD problem on a PC was studied. With coarse grids, an excellent speedup of 350% was achieved, while for fine grids the speedup was 256%. In future work, processors with larger core numbers as well as graphics card based solutions will be investigated to establish their associated speedups in newer PCs. The reduction in turnaround times with the Euler code will make it attractive in aeroelasticity analysis work, where a large number of simulations need to be executed; improving the accuracy of aerodynamic computations (compared with the full potential and transonic small disturbance equations) in the preliminary design of aircraft wings; or can even be used as an educational tool for students studying computational aerodynamics.

Acknowledgements The authors thank RMIT University, Melbourne, Australia, for supporting this research work through a research grant, RMIT Emerging Researcher Grant 2007, and the School of Mathematical and Geospatial Sciences of RMIT University for providing the necessary computer software and hardware.

References

[1] Advanced Micro Devices (AMD), *AMD FireStream 9170: Industry's First GPU with Double-Precision Floating Point*, 2007, http://ati.amd.com/products/streamprocessor/specs.html C578

[2] Allmaras, S. R., Venkatakrishnan, V., and Johnson, F. T., *Farfield Boundary Conditions for 2-D Airfoils*, AIAA Paper 2005-4711, AIAA, 2005. C582

[3] Bailey, D., Barszcz, E., Barton, J., Browning, D., Carter, R., Dagum, L., Fatoohi, R., Fineberg, S., Frederickson, P., Lasinski, T., Schreiber, R., Simon, H., Venkatakrishnan, V., and Weeratunga, S., *The NAS Parallel Benchmarks*, National Aeronautics and Space Administration (NASA), Contractor Report 203186, USA, 1994,

http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19970014936_1997021577.pdf C577

[4] Dong, S., and Karniadakis, G. E., Dual-Level Parallelism for High-Order CFD Methods, *Parallel Computing*, **30**, Jan. 2004, pp. 1–20. C577

[5] Hirsch, C., *Numerical Computation of Internal and External Flows (Volume 2): Computational Methods for Inviscid and Viscous Flows*, John Wiley and Sons, Great Britain, 1992. C580, C581, C582

[6] Hoffmann, K. A., *Computational Fluid Dynamics for Engineers*, Engineering Educational System, Texas, USA, 1989. C578, C580

[7] Intel, *Tera-scale Computing Research Program*, c.2007, http://www.intel.com/research/platform/terascale/ C578

[8] Jin, H., Frumkin, M., and Yan, J., *The OpenMP Implementation of NAS Parallel Benchmarks and its Performance*, National Aeronautics and Space Administration (NASA), Technical Report NAS-99-011, Moffett Field, USA, 1999. C577, C585

[9] Liu, Y., and Vinokur, M., *Nonequilibrium Flow Computations 1: An Analysis of Numerical Formulations of Conservation Laws*, National Aeronautics and Space Administration (NASA), Contractor Report 177489, California, USA, 1988, http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19880021000_1988021000.pdf C581

[10] Lock, R. C., *Test Cases for Numerical Methods in Two-Dimensional Transonic Flows*, Advisory Group for Aerospace Research and Development (AGARD), Report Number 575, 1970. C584

[11] Ly, E., Improved Approximate Factorisation Algorithm for the Steady Subsonic and Transonic Flow over an Aircraft Wing, *Proceedings of the 21st Congress of the International Council of the Aeronautical Sciences*

(ICAS98), ICAS and AIAA, Melbourne, Australia, 1998, Paper A98-31699. C578, C579

[12] Ly, E., and Gear, J. A., Time-Linearized Transonic Computations Including Shock Wave Motion Effects, *Journal of Aircraft*, **39**, 6, Nov/Dec. 2002, pp. 964–972. C578, C579, C585

[13] Ly, E., and Nakamichi, J., Time-Linearised Transonic Computations Including Entropy, Vorticity and Shock Wave Motion Effects, *The Aeronautical Journal*, Nov. 2003, pp. 687–695. C578, C579, C585

[14] Ly, E., and Norrison, D., Automatic Elliptic Grid Generation by an Approximate Factorisation Algorithm, *ANZIAM Journal*, **48** (CTAC2006), pp. C188–C202, July 2007, http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/48. C578, C579

[15] Ly, E., and Norrison, D., Generating Elliptic Grids in Three Dimensions by a Method of False Transients, *ANZIAM Journal*, **49** (EMAC2007), pp. C170–C183, Nov. 2007, http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/313. C578, C579

[16] Nvidia, *The Era of the Personal Supercomputer*, 2007, http://www.nvidia.com/content/events/siggraph_2007/supercomputing.html C578

[17] Pope, S. B., *A Perspective on Turbulence Modeling*, Modeling Complex Turbulent Flows, Kluwer Academic Publishers, The Netherlands, 1999. C585

[18] Pulliam, T., Kutler, P., and Rossow, V., *Harvard Lomax: His Quiet Legacy to Computational Fluid Dynamics*, 14th AIAA Computational Fluid Dynamics Conference, Norfolk, VA, 28 June-1 July, 1999, http://people.nas.nasa.gov/~pulliam/mypapers/lomax_la.ps C577

[19] Steger, J. L., and Warming, R. F., *Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite Difference Methods*, National Aeronautics and Space Administration (NASA) Technical Memorandum 78605, California, USA, 1979, http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19790020779_1979020779.pdf C581

[20] Thompson, J. R., Warsi, Z. U. A., and Mastin, C. W., *Numerical Grid Generation: Foundations and Applications*, Elsevier Science Publishing Co. Inc., New York, USA, 1985. C582

[21] Van Leer, B., Flux-Vector Splitting for the Euler Equations, *Proceedings of the 8th International Conference on Numerical Methods in Fluid Dynamics*, Aachen, West Germany, 1982. C578, C581

Author addresses

1. **D. Norrison**, School of Mathematical and Geospatial Sciences, RMIT University, Melbourne, Victoria 3001, Australia.
   mailto:daniel.norrison@rmit.edu.au

2. **E. Ly**, School of Mathematical and Geospatial Sciences, RMIT University, Melbourne, Victoria 3001, Australia.