

Computational Fluid Dynamics using OpenCL – a Practical Introduction

T Bednarz¹, Luke Domanski² and J A Taylor³

¹ *CSIRO Mathematics Informatics & Statistics, Sydney, Australia.*
Email: tomasz.bednarz@csiro.au

² *CSIRO Advanced Scientific Computing, Sydney, Australia*

³ *CSIRO Mathematics Informatics & Statistics, Canberra, Australia*

Abstract: The main aim of the Computational Fluid Dynamics (CFD) simulations is to reconstruct the reality of fluid motion and behaviour as accurately as possible in order to better understand the natural phenomena under specified conditions. Ideally, general solutions can also cover different scales and geometric configurations. Unfortunately, due to expensive algorithms, classic CFD codes most often require long computational times to satisfy the convergence criteria. With the advent of high-performance GPUs with massively-parallel multi-threaded architectures, basic CFD algorithms can now be implemented to give results in near real-time. The current work will briefly review our existing explicit solver based on finite difference methods, the derivation and discretisation of the mathematical model and equations, through to GPU algorithm implementation. During presentation, several case studies computed using CSIRO's CPU/GPU supercomputer cluster will be described and compared against well known analytical and experimental solutions, i.e. natural convection, driven cavity, scaling analysis, magneto-thermal convection, etc.

Keywords: *Computational Fluid Dynamics, Simulations, Numerical Modelling*

1. INTRODUCTION AND MODEL EQUATIONS

Experiments with fluids are usually very expensive and many times not feasible, or require work in harmful environments, etc. Therefore, in such cases, we're trying to replace them with numerical modelling whenever possible. Here comes Computational Fluid Dynamics (CFD) that aims to reconstruct the reality of fluid motion and behaviour as accurately as possible in order to better understand the natural phenomena under specified conditions. Ideally, general solutions can also cover different scales and geometric configurations and should be with an agreement with equivalent experimental results.

Unfortunately, due to expensive algorithms, classic CFD codes most often require long computational times to satisfy the convergence criteria. With the advent of high-performance GPUs with massively-parallel multi-threaded architectures, basic CFD algorithms can now be implemented to give results in near real-time. The algorithms used to solve these problems utilize GPU and in this case OpenCL. Presented results demonstrate that GPU can be successfully used to accelerate fluid simulations. We have seen significant gains in productivity and opportunity as a result of leveraging GPUs, being able to tackle computational problems in which execution time was previously infeasible.

1.1. Model equations

In order to solve general incompressible thermo-fluids problems in Cartesian coordinate system, the Navier-Stokes and energy equations can be defined as follows:

$$\vec{\nabla} \cdot \vec{U} = 0 \quad (1)$$

$$\frac{D\vec{U}}{Dt} = -C_1 \vec{\nabla} P + C_2 \nabla^2 \vec{U} + C_3 (\theta - C_4) \quad (2)$$

$$\frac{D\theta}{Dt} = C_5 \nabla^2 \theta \quad (3)$$

where \vec{U} is the velocity, t is the time, P is the pressure, θ is the temperature and coefficients C_1 to C_5 will depend on a specific case considered. The same methodology can be applied to simulate wide range of different fluid flows: pipe flows, force convections, magneto-thermal convection, scaling analysis, exchange flows in reservoir model, mixing, fountain flows, bubble flow, step flows, heat exchangers, ventilation problems, etc. For instance, if under the consideration is a square cavity heated from one vertical wall and cooled from the opposite one with top and bottom walls kept adiabatic, those coefficients for dimensionless solution are defined as follows:

$$C_1 = 1; C_2 = Pr = \frac{\nu}{\alpha}; C_3 = Pr \cdot Ra = \frac{\nu}{\alpha} \cdot \frac{g\beta(\Delta\theta)l^3}{\alpha\nu}; C_4 = 0; C_5 = 1.0 \quad (4)$$

where ν is the kinematic viscosity, α is the thermal diffusivity, g is the gravitational acceleration, β is the thermal expansion coefficient, $\Delta\theta$ is the maximum temperature difference, l is the length of the cube, Pr is the Prandtl number describing ratio of momentum and thermal diffusivities and Ra is the Rayleigh number associated with buoyancy driven flow describing strength of the convection.

2. NUMERICAL APPROACH

Those equations are approximated with finite difference equations and the HSMAC (Highly Simplified Marker and Cell) method (Bednarz *et al.* 2005-2010, Hirt 1975) is used to iterate mutually the pressure and velocity fields on staggered mesh/grid allocation system, see Figure 1. The inertial terms in momentum equations are approximated using a third-order upwind UTOPIA scheme (Tagawa 1996). The absolute convergence criteria for the numerical solutions are specified based on the residual sums of all conserved quantities. If the residual sum is less than 10^{-6} for each conserved quantity, the equations are deemed to have converged at a specific time step. The time-step is chosen to ensure numerical stability according to the CFL condition. The numerical methods used in this work for simulation of natural convection have been widely verified by co-workers, by both the numerical and the experimental investigations for closely related problems (Bednarz *et al.* 2005-2010).

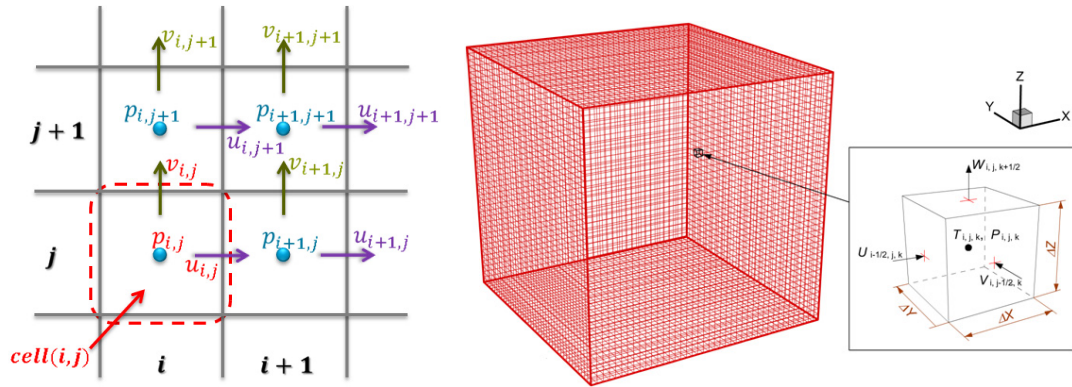


Figure 1. Staggered mesh grid allocations prevents possible pressure oscillations. Left: for 2-D cases all scalar variables are defined in the centre of cells, and vector variables in at the centre of vertical and horizontal cell faces accordingly. Right: grid for 3-D cases.

3. OPENCL IMPLEMENTATION

The numerical code is ported from CPU to GPU version using the OpenCL API, see references. This was motivated by the need of improving computation speed, as in some cases, e.g. computation of single case of boundary layer evolution (Bednarz *et al.* 2009) could take up to 12 hours on grid size 256x256. Therefore, all critical parts of previously available CPU code are re-implemented in several OpenCL kernels that could be executed by thousands simultaneous threads by a GPU. Figure 2 shows the flow chart of our HSMAC-OpenCL solver. As seen, the initialization part includes: reading initial configuration files describing geometry and parameters of the problem to be computed, allocating memory for all field variables (pressure, velocity components, temperatures and spatial coordinates), preparing boundary conditions flags (to mark regions where the boundaries are located and what is their type). Once that's done, the OpenCL is initialized, proper compute device is attached to its context and the CL program is compiled. Also the device memory buffers are created and filled with initial data.

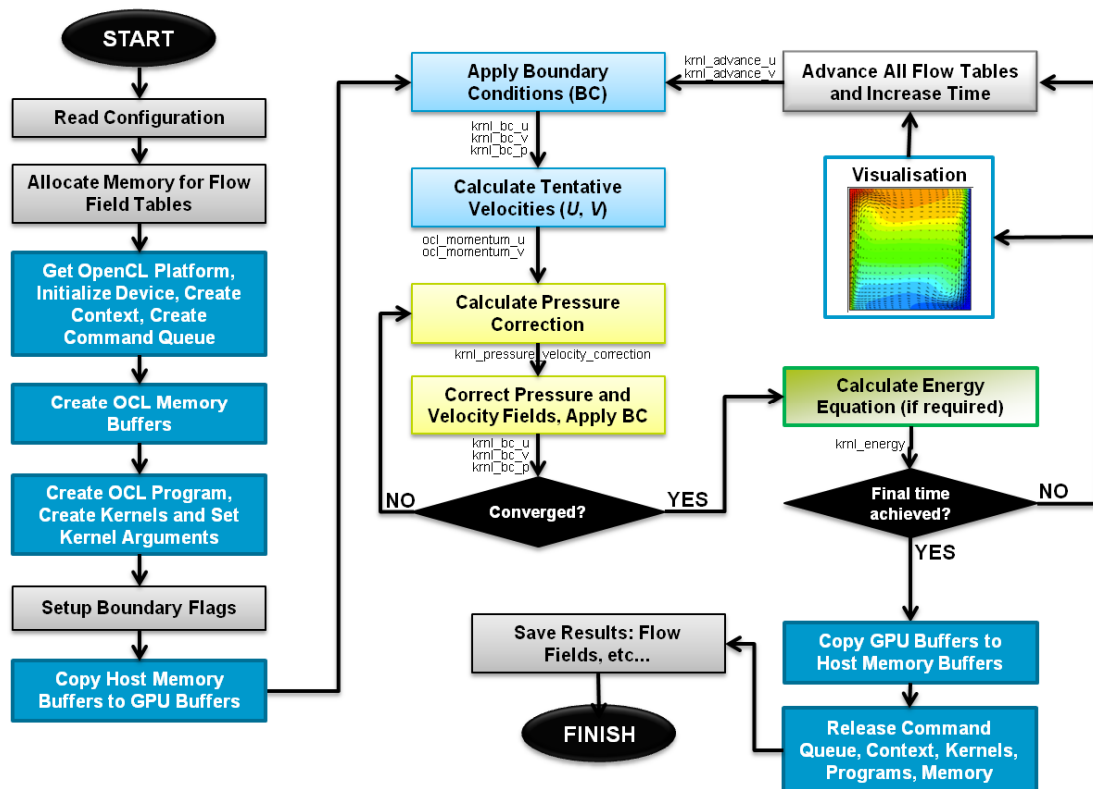


Figure 2. Flow char of the OpenCL solver.

The solver / runtime execution of all OpenCL kernels is controlled by host device (CPU). The flow chart shows two main loops: the outer loop, which is responsible for general time-step iterations and the inner loop used for mutual iteration of velocity and pressure fields to satisfy the continuity equation, before solving each energy equation time-step (Hirt 1975). For instance calculation of pressure and velocity correction is done by execution of the kernel *kml_pressure_velocity_correction*, calculation of energy equation by *kml_energy*, etc. In addition the interoperability feature of OpenCL with OpenGL can be used to visualize the results while they are still under computation. After reaching final time step, all results can be saved on disk for further analysis.

Figure 3 shows simplified code snippet of sample OpenCL kernel being used to calculate vertical component of tentative velocity from momentum equation. Please notice, that all derivatives for vertical velocity are calculated at the middle of top vertical cell edge. For simplicity of presentation, convection acceleration is calculated using central approach. In real simulations, as mentioned UTOPIA was used for accuracy and stability.

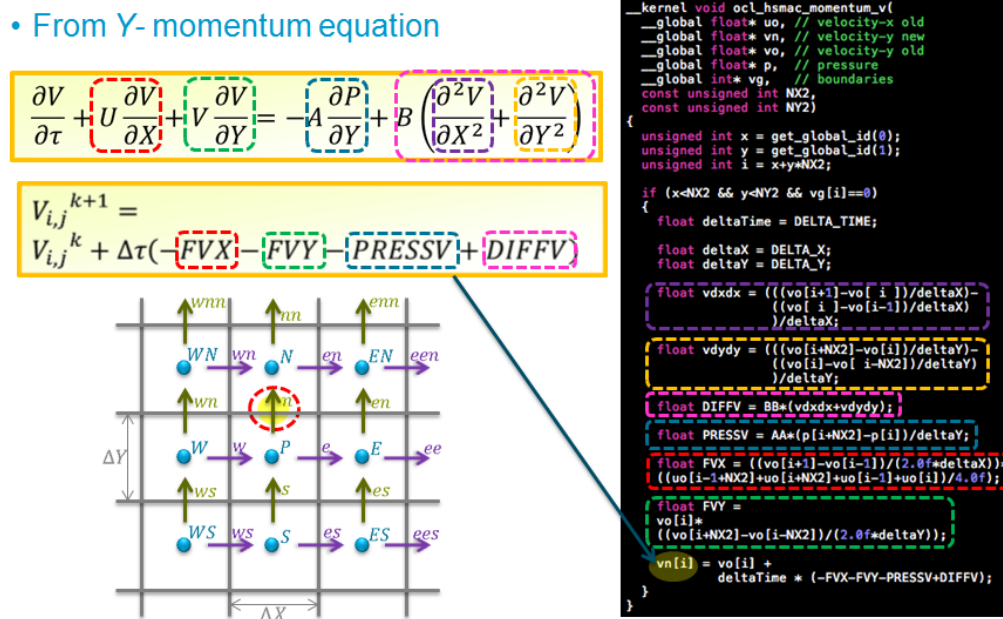


Figure 3. Simplified OpenCL kernel that calculates tentative, vertical velocity component from the momentum equation.

4. RESULTS AND CONCLUSIONS

Figure 4 depicts total execution times to reach final time of $t = 5$ for natural convection simulations at $Pr = 0.71$ and $Ra = 10^6$ versus different grid sizes denoted in total cell numbers (product of grid sizes in X - and Y -direction). These results show potential speed-ups that can be achieved using GPU compute devices. Initial tests comparing GT 330M to Intel Core i7 showed average 30X speedup of GPU/CPU. Further, comparing S2050 execution times to GT 330M, there was additional ~ 7 -9X speedup (which is approximately equivalent to the ratio of the number of streaming cores between those two GPUs). These results are indication of what can be achieved when using OpenCL with today's GPUs for solving explicitly the Navier-Stokes equation for natural convection flows, i.e. obtaining converged results in few minutes instead of 10-12 hours.

5. REFERENCES

Bednarz, T., Tagawa, T., Kaneda, M., Ozoe, H., and Szmyd, J.S. (2005) The convection of air in a cubic enclosure with an electric coil inclined in general orientations, *Fluid Dynamics Research*, Vol. 36, No. 2, pp. 91-106.

Bednarz, T., Lin, W., Patterson, J.C., Lei, C., and Armfield, S.W. (2009) Scaling for unsteady thermo-magnetic convection boundary layer of paramagnetic fluids of $Pr > 1$ in microgravity conditions, *International Journal of Heat and Fluid Flow*, Vol. 30, Issue 6, pp. 1157-1170.

Bednarz, T., Caris, C., and Taylor, J. (2010) A practical introduction to Computational Fluid Dynamics on GPUs, GTC 2010, San Jose, USA, September 20-23.

Botella, O., and Peyret, R. (1998) Benchmark spectral results on the lid-driven cavity flow, *Computers & Fluids*, Volume 27, Issue 4, Pages 421-433.

CSIRO GPU cluster: <http://www.csiro.au/resources/GPU-cluster.html>.

De Vahl Davis, G. (1983) Natural convection of air in a square cavity: a bench mark numerical solution, *International Journal of Numerical Methods in Fluids*, Vol 3, pp 249-264.

Hirt, C.W., Nichols, B.D., and Romero, N. (1975) A numerical solution algorithm for transient fluid flow, Los Alamos Scientific Laboratory, LA-5852.

OpenCL: <http://www.khronos.org/opencl/>

Tagawa, T., and Ozoe, H. (1996) Effect of Prandtl number and computational schemes on the oscillatory natural convection in an enclosure, *Numerical Heat Transfer A* 30, 271-282.

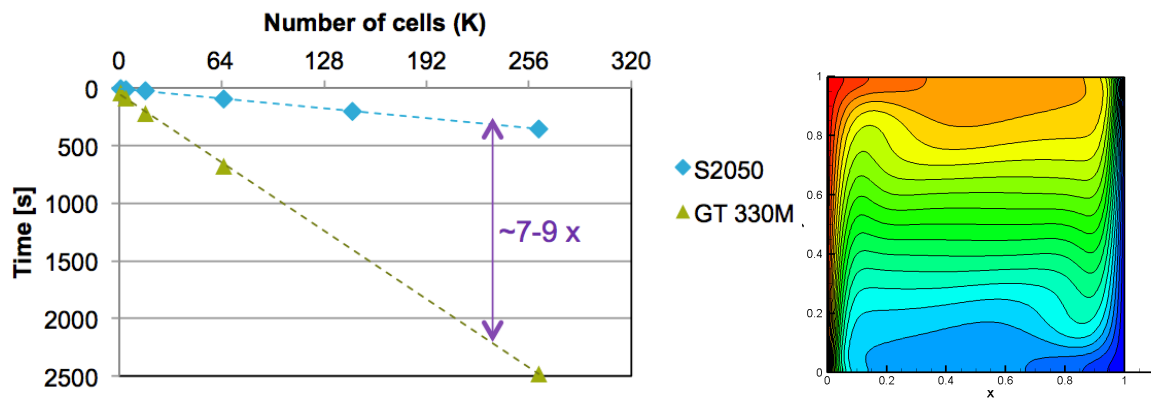


Figure 4. Total execution times for S2050 and GT330M against number of computational cells and converged temperature contours for $Pr = 0.71$ and $Ra = 10^6$ (left wall = heated, right wall = cooled).