Acceleration of a Finite-Difference WENO Scheme for Large-Scale Simulations on Many-Core Architectures

Athanasios S. Antoniou † Konstantinos I. Karantasis † Eleftherios D. Polychronopoulos ‡ and John A. Ekaterinaris *

University of Patras, 26500 Patra, Greece

Current trends on high performance computing are moving towards the deployment of several cores on the same chip of modern processors in order to achieve substantial execution speedup through the extraction of the potential fine-grain parallelism of applications. At the forefront of this trend we find nowadays the modern Graphics Processors Units (GPUs), which due to their simplistic design are able to encompass hundreds of independent processing units on a single chip in contrast to their respective CPUs, which at the moment include only a few cores on the same chip. In order to study the potential speedup of computationally intensive applications that utilize the many-core architecture of GPUs, this paper presents a highly accelerated implementation of the finite-difference weighted essentially non-oscillatory (WENO) scheme. This method is suitable for direct numerical simulations (DNS) large eddy simulations (LES) of compressible turbulence and requires large computing resources in order to achieve high Reynolds numbers. Our implementation targets on large-scale simulations using the CUDA parallel programming and constitutes a paradigm of GPU’s applications in CFD. The results of the current implementation demonstrate that such a computationally intensive application could be highly accelerated running on the NVIDIA Tesla C1070 many-core GPU.

I. Introduction

NUMERICAL simulation of transition and turbulence in high-speed flows is daunting because of the difficulty in ensuring high-resolution and fidelity in capturing small disturbances in an environment containing sharp gradients associated with shocks and relatively thin boundary layers. Even with use of higher-order approaches many shock capturing methods introduce spurious (numerical) noise which contaminates the solution beyond acceptable limits, which can lead to significant damping of turbulence fluctuations, and masks the effects of the subgrid-scale (SGS) models. Specification of boundary conditions ensuring that the numerical discretizations remain stable is also a critical issue. Robust, high-fidelity and accuracy methodologies that are capable of treating complex flows and are applicable for high-resolution numerical solutions in complex domains are therefore solemnly required.

The objective of the present work is to apply a conservative, high-order accurate, shock-capturing method that is suitable for the simulation of supersonic flows in domains with moderate complexity for massively parallel architectures. A high order finite difference weighted essentially nonoscillatory (WENO) scheme is used for the numerical solution of the governing equations. The numerical method is based on high order discretizations applied to the curvilinear coordinate formulation of the compressible Navier-Stokes equations. The governing equations, the subgrid scale model used for turbulence simulations, and high order WENO schemes for the discretization of the inviscid fluxes are presented in the following sections. The main disadvantage of this method is that with the increase

† Graduate Student, School of Computer Engineering and Informatics.
‡ Assistant professor, School of Computer Engineering and Informatics.
* Professor, School of Mechanical and Aerospace Engineering, Associate Fellow AIAA.
of the order of accuracy increases the width of the computational stencil. As a result, parallelization with traditional domain decomposition methods could become inefficient. In this paper we demonstrate acceleration with the use of graphics processors units (GPUs).

The implementation of the accelerated finite difference WENO scheme that we present here is based on the CUDA parallel programming model. CUDA comprises a programming environment that provides an extension to the C programming language accompanied by the necessary libraries to support the execution of code on top of the NVIDIA GPUs. Using CUDA the program must be structured on distinct portions which are called kernels and are destined to be executed on the side of the GPU. The execution of kernels follows the programming paradigm of SIMT (Single Instruction Multiple Threads) which resembles the traditional programming model of vector processors, the SIMD (Single Instruction Multiple Data) paradigm. Nevertheless SIMT is less restrictive than SIMD and allows programmers to write either data-parallel code as in the case of SIMD or arbitrary thread-based parallel code. In order to initiate a kernel execution using CUDA a mapping of application threads into blocks and accordingly a mapping of these blocks on a grid must be provided by the programmer.

Data parallel applications are serious candidates among the applications that have the potential to achieve a significant acceleration through the utilization of many-core GPUs. Similar efforts have recently taken place in the area of scientific and recent demonstrations in CFD concern simulations on both structured and unstructured meshes. The work that we present here proves that CFD applications with similar characteristics can benefit from heterogeneous computing systems that involve CPU and GPU co-processing.

II. Numerical Implementation

A. Governing equations

The Favre-averaged Navier-Stokes equations are solved numerically. Favre averaging corresponds to a density-weighted filtering operator \( \hat{\phi} = \rho \phi / \rho \) for every flow variable, \( \phi \), which is decomposed as \( \phi = \hat{\phi} + \phi'' \), where \( \phi'' \) represents mass-weighted subgrid fluctuations. The Cartesian coordinate form of the Favre-averaged continuity momentum and energy equations is:

\[
\frac{\partial \hat{\rho}}{\partial t} + \frac{\partial (\hat{\rho} \hat{u}_j)}{\partial x_j} = 0 \quad (1.1)
\]

\[
\frac{\partial (\hat{\rho} \hat{u}_j)}{\partial t} + \frac{\partial (\hat{\rho} \hat{u}_j \hat{u}_j)}{\partial x_j} = -\frac{\partial \hat{p}}{\partial x_j} + \frac{\partial \hat{\tau}_{ij}}{\partial x_j} + \frac{\partial \hat{\epsilon}_{iS}}{\partial x_j} \quad (1.2)
\]

\[
\frac{\partial (\hat{\rho} \hat{E})}{\partial t} + \frac{\partial (\hat{\rho} \hat{E} + \hat{p} \hat{u}_j \hat{u}_j)}{\partial x_j} = -\frac{\partial}{\partial x_j} \left( c_v \frac{\partial \hat{T}}{\partial x_j} \right) + \frac{\partial}{\partial x_j} \left[ \hat{u}_j \left( \hat{\tau}_{ij} + \hat{\epsilon}_{iS} \right) \right] \quad (1.3)
\]

where \( \hat{E} = \hat{p} \gamma / (\gamma - 1) + 0.5 \hat{\rho} \hat{u}_j \hat{u}_j \) is the computable energy and \( \hat{\tau}_{ij} \) is the Favre-filtered viscous stress tensor,

\[
\hat{\tau}_{ij} = \mu \left( 2 \hat{S}_{ij} - \frac{2}{3} \hat{S}_{kk} \delta_{ij} \right) \quad (2)
\]

with \( \hat{S}_{ij} = 0.5 (\hat{\rho} \hat{u}_i / \hat{\rho} \hat{x}_i + \hat{\rho} \hat{u}_j / \hat{\rho} \hat{x}_j) \), and the molecular viscosity \( \mu \) is computed from the filtered temperature using Sutherland’s law

\[
\mu(\hat{T}) = \hat{T}^{\frac{1}{2}} \left[ \frac{1 + 0.76}{\hat{T} + 0.76} \right] \quad (3)
\]

On the right hand side of the energy equation, Eq. (1.3), the additional diffusive terms exist
\[-\frac{\partial}{\partial x_j} (\gamma C_i Q_j + J_i - D_i) \tag{4}\]

where the terms in Eq. (4) are given by:

\[Q_j = \overline{\rho} (\overline{u_i T} - \overline{u_i \overline{T}})\]
\[J_j = \overline{\rho} (\overline{u_i u_k} - \overline{u_i u_k})\]
\[D_j = \tau_{ij} u_i - \tau_{ij} u_i\] \tag{5}

Vreman et al.\(^5\) performed a priori test using DNS data obtained from the calculation of a compressible mixing layer, and concluded that these nonlinearities have negligible effects and it is acceptable to neglect them. For high Mach number LES, it is not absolutely clear that the contribution of these terms is still negligible. However, because a suitable subgrid scale model is not available for these terms. The subgrid-scale viscous tensor \(\tau_{ij}^{SGS}\) is modeled by:

\[\tau_{ij}^{SGS} = \mu_{\text{tur}} \left( 2 \overline{\delta_{ij}} - 2 \overline{S_{ik} S_{kj}} \right) - 2 \overline{\rho} k_{SGS} \overline{S_{ij}} \tag{6}\]

In the modeled subgrid-scale tensor \(\tilde{\tau}_{ij}^{SGS}\) of Eq. (6), the eddy viscosity \(\mu_{\text{tur}}\) and \(k_{SGS}\) depend on the subgrid scale model that is presented in the next section and in Eq. (1.3) the subgrid heat flux \(q_{SGS}^{\prime}\) is modeled by:

\[q_{SGS}^{\prime} = -c_p \frac{\mu_{\text{tur}}}{\text{Pr}_{\text{tur}}} \frac{\partial \overline{T}}{\partial x_i} \tag{7}\]

The simulations of the present investigation have been carried out in very simple domains. The finite difference meshes along the stream wise and span wise directions were uniform, while along the normal to the wall direction simple stretching was used. For these meshes, it is not absolutely necessary to use curvilinear transformations of coordinates. However, in order to develop a more general computational approach a curvilinear transformation of coordinates was employed.

**B. Subgrid scale model and constants**

One of the main uncertainties in LES is the subgrid scale parameterization. Development of subgrid models that attempt to faithfully predict the dynamics of the interaction between the resolved and unresolved scales, especially for high speed flows, is an active area of research. In the current implementation, the compressible version of the Smagorinsky subgrid-scale model\(^6\) with a fixed filter width is used. This model, similar to most existing subgrid scale models for LES of high speed flows, is based on the assumption that only the compressibility effects associated with the variations of mean quantities are significant and those associated with turbulent fluctuations are negligible. The values of \(\mu_{\text{tur}}\) and \(k_{SGS}\) in this model are evaluated as

\[\mu_{\text{tur}} = C_s \rho \Delta^2 \sqrt{\overline{S_{ij} S_{ij}}}\]
\[k_{SGS} = C_f \Delta^2 \overline{S_{ij} S_{ij}}\] \tag{8}

where the constants values \(C_R = 0.012\) and \(C_f = 0.0066\) are used and the filter width is chosen as \(\Delta = (\Delta x \times \Delta y \times \Delta z)^{1/3}\). It is well known that the Smagorinsky subgrid-scale model is not very accurate for wall bounded flows. The deficiencies of this model can be improved when a length scale filter close to the wall is introduced. A modified version the Smagorinsky subgrid-scale model was proposed by Nicoud and Ducros\(^7\) to overcome the shortcomings encountered for wall bounded flows. Other improved subgrid scale models for wall bounded flows were proposed in Refs. 8 and 9.
C. Numerical scheme

The Favre-averaged Navier-Stokes equations were written in generalized coordinates. The discretization of the inviscid fluxes is based on standard finite difference WENO scheme. The numerical code includes options for 5th, 7th and 9th order accurate discretizations of the inviscid fluxes. The viscous fluxes were evaluated with a 4th order accurate explicit, central difference scheme by evaluating the second derivatives with repeated evaluation of the first derivative, first at half points and then at the nodes of the finite difference mesh. High order accurate discretization of the viscous fluxes requires large computational time. It was found that for a fourth order accurate explicit discretization, significant portion of computational time per time step is spent for the evaluation of the viscous fluxes and that higher order explicit or compact discretizations are prohibitively expensive computationally.

The classical tree stage, third order accurate TVD Runge-Kutta method of Osher and Shu is used for time marching. For LES computations the Smagorinsky subgrid scale model is used.

The essential details of the finite difference WENO discretization for an equally spaced mesh are given next. The reconstruction by WENO uses a convex combination of k candidate ENO stencils, $S_q(j) = \{x_{j-q}, \ldots, x_{j-1}\}$, $q = 0, \ldots, k-1$. Each ENO stencil $S_q(j)$ produces a $k$th order accurate ENO reconstruction $f_{j+1/2}^{(q)} = \sum_{m=0}^{k-1} c_{q,m} \bar{f}_{j+1/2-m}$, where for the finite difference formulation is the nodal value $\bar{f}_{j+1/2-m}$. The convex sum of the WENO reconstruction

$$\bar{f}_{j+1/2} = \sum_{q=0}^{k-1} \omega_q f_{j+1/2}^{(q)} = f_{j+1/2} + O(\Delta x^{2k}), \quad \omega_q \geq 0, \quad \sum_{q=0}^{k-1} \omega_q = 1$$

produces a $(2k-1)$th order approximation at the cell boundaries of an equally spaced mesh, which is the transformed domain mesh. For example the reconstruction of a left state $f_L = \bar{f}_{j-1/2}$ about $j-1/2$ for the fifth order accurate WENO scheme is obtained as

$$f_L = \omega_0 q_0 + \omega_1 q_1 + \omega_2 q_2,$$  \hfill (10)

where

$$q_0 = \frac{1}{3} f_{j-2} - \frac{7}{6} f_{j-1} + \frac{11}{6} f_j,$$

$$q_1 = -\frac{1}{6} f_{j-1} + \frac{5}{6} f_j + \frac{1}{3} f_{j+1},$$

$$q_2 = \frac{1}{3} f_j + \frac{5}{6} f_{j+1} - \frac{1}{6} f_{j+2},$$  \hfill (11)

The nonlinear weights $\omega_q$ in Eq. (9) are:

$$\omega_q = \frac{\alpha_q}{\sum_{q=0}^{k-1} \alpha_q}, \quad \alpha_q = \frac{d_q}{(\varepsilon + \beta_q)}, \quad \sum_{q=0}^{k-1} d_q = 1$$

where $d_q$ are positive constants (optimal weights) for the smooth stencils, $\beta_q$ are the smoothness indicators of the stencil $S_q(j)$, and $\varepsilon > 0$ is a constant to avoid division by zero. The value of this constant was taken $\varepsilon = 10^{-6}$ for all computations. The smoothness indicators $\beta_q$ were computed as sum of the squares of $L^2$ norms of the derivatives of the interpolation polynomial. Smoothness measures for the fifth order accurate WENO reconstruction of Eq. (10) are defined as
\[ \beta_0 = \frac{13}{12} (f_{i-2} - 2f_{i-1} + f_i)^2 + \frac{1}{4} (f_{i-2} - 4f_{i-1} + 3f_i)^2 \]
\[ \beta_1 = \frac{13}{12} (f_{i-1} - 2f_{i-1} + f_{i+1})^2 + \frac{1}{4} (f_{i-1} - 4f_i + 3f_{i+1})^2 \]
\[ \beta_2 = \frac{13}{12} (f_i - 2f_{i+1} + f_{i+2})^2 + \frac{1}{4} (f_{i-2} - 4f_{i-1} + 3f_{i+2})^2 \]  

(13)

The reconstruction of the right state about \( j+1/2 \) is symmetric. Smoothness measures for \( k = 3, 4, 5 \) or \( (2k-1)-th \), 5th, 7th, and 9th order accurate WENO reconstructions are given in Ref. 12.

High order finite difference WENO discretization of the inviscid fluxes clearly requires wide stencils. For example the 5th order accurate WENO scheme requires a seven point wide stencil while the 9th order accurate one involves an eleven point wide stencil. These wide stencils may present problems for the specification of boundary conditions and make less efficient the parallelization with domain decomposition techniques. In this work, boundary conditions were specified using the ghost cell approach for all boundaries. Furthermore, the simplest Lax-Friedrichs splitting \( f'(u) = 0.5(f(u) \pm a u) \), \( a = \max |f'(u)| \) is used for the evaluation of the numerical flux.

For the numerical solution of the three-dimensional Euler or Navier-Stokes system with the finite difference WENO scheme it is necessary to define an average state of the left, \( f_L \) and the right \( f_R \) states at an interface. A Roe-type approximate Riemann solver is used. The Lax-Friedrichs numerical flux is used and the average state \( \overline{U} \) is defined for the variables \( U = [\rho, u, v, w, p]^T \) using Roe's average as

\[ \overline{\rho} = \sqrt{\rho_L \rho_R}, \quad \overline{u} = \frac{u_L \sqrt{\rho_L} + u_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \]
\[ \overline{p} = \frac{p_L \sqrt{\rho_L} + p_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \]  

(14)

Once the average state has been defined the conservative flux vectors are first projected on the characteristic space where the reconstruction is carried out and then the numerical flux is re-projected back to the conservative variables space and the derivatives are computed as

\[ \frac{\partial \hat{f}}{\partial \zeta} = \frac{\hat{f}_{j+1/2} - \hat{f}_{j-1/2}}{\Delta \zeta} \quad \frac{\partial \hat{f}}{\partial x} = \frac{\hat{f}_{j+1/2} - \hat{f}_{j-1/2}}{\Delta x} \]  

(15)

where \( \hat{f}_{j+1/2} \) denotes the numerical flux which is reconstructed using high order accuracy. Therefore the evaluation of a proper set of right and left eigenvectors for the conservative flux vectors is a crucial step of the WENO solver.

A curvilinear coordinates transformation was applied and finite difference discretizations were applied in the equally space transformed domain. The metric quintiles were computed with the standard compact fourth order finite difference formulas. It was found that due to the simplicity of the mesh evaluation of metrics with sixth order accurate formulas does not make any difference.

### III. Acceleration on Graphics Processors

#### A. Domain decomposition

Similar to other computational techniques used in the CFD community, the high order WENO scheme for the numerical solution of the Euler and Navier-Stokes equations, has been initially implemented in FORTRAN-90 for single block structured meshes. Parallelization of the code for clusters was subsequently achieved through the use of the message passing protocol MPI and domain decomposition. On the MPI parallel version, the domain decomposition was performed by subdividing the domain only along the axial direction. In order to study the
potential acceleration of the code on heterogeneous schemes that involve CPU and GPU co-processing we decided to port our application on the C language although the first parallelizing compiler for CUDA and Fortran have been released by the time that our research was taking place. According to that decision, the FORTRAN-90 parallelized version of the code formed the basis for the shared memory parallel implementation on C/OpenMP that we evaluate in this paper. The decomposition of the computational domain for OpenMP multi-threaded execution is depicted in the schematic of Fig. 1. The generalized coordinates form of the governing equations is solved and the global computation of dimension $I_{max} \times J_{max} \times K_{max}$ is subdivided along the $i$, or $\xi$ direction, which is often the streamwise direction, in $N$ subdomains of dimension $(\lfloor I_{max}/N \rfloor + 2m) \times J_{max} \times K_{max}$, where $m$ is the number of the ghost points required for data transfer. The number of ghost points varies with the order of the base scheme and for the 5th order WENO is $m=3$, while for the 9th order WENO is $m=5$.

Figure 1. Schematic for domain decomposition with MPI or OpenMP

The C/OpenMP version of the WENO solver was validated by comparing results of the baseline FORTRAN-90 code for several cases for both single processor solutions and solutions with domain decomposition. Next parallel implementation on GPUs was implemented and the C language version was ported into CUDA.

B. GPU processing

In the presence of considerably larger number of cores on the GPU as opposed to the CPU, in order to fully utilize the afforded computational resources, we extended the domain decomposition along a second direction. In that sense every element on the 2D plane that is mapped on the GPU is assigned on a different thread and corresponds to the processing of a group of points along the third axis. This strategy of a 2D domain decomposition that is depicted in Fig. 2 and is exercised both on the single GPU mode and the multi-GPU mode adds additional capabilities for large scale DNS or LES simulations where large number of points is used along the streamwise and normal to the wall directions while the number of points in the spanwise direction, which is often considered periodic, is smaller. Moreover, to maximize the throughput with appropriate expression of parallelism, the number of threads on each block that handles the respective points has been set as a multiple of 32, which corresponds to the warp size of the GPU. This was decided because the warp is the minimum unit that can be scheduled at once on a single stream multiprocessor of the GPU.
Along with the need to express a high degree of parallelism on our scheme, the parameter that we considered and has a great impact on performance is the memory utilization, especially in the case of multi-GPU execution. For that reason, firstly, we had to minimize data transfers between the host memory and the device memory. Therefore, in the presented implementation all the computations that are required by the Runge Kutta time stepping for updating the numerical solution have been assigned to the GPU, even though some of the computations do not exhibit a high degree of parallelism.

Subsequently on the side of each GPU, in order to utilize appropriately the memory hierarchy of the device, we use the texture memory space for the placement of read-only data that are computed before the time evolution of the simulation begins. This happens in order to benefit from the caching mechanism of the texture memory, which is by design optimized for spatial locality. The results that are computed on each time step are stored in global – read/write – memory. In addition, due to the specification of CUDA architecture that forbids communication between threads that reside on distinct blocks, each block computes its ghost points instead of exchanging them with neighboring blocks.

Concerning the necessary synchronization that is imposed by the computation scheme, it is restricted on barrier synchronization between the threads of the same common block inside each CUDA kernel. In that way we signify the update of auxiliary local variables required for Roe’s averaging and other local arrays for the right and the left eigenvectors that are evaluated at the average state. The need for mutual exclusion is minimized on the atomic max operation that is used in order to compute the maximum eigenvalue required for the construction of the Lax-Friedrichs numerical flux.
IV. Results

A. Computational examples

Two examples were used to evaluate the performance of the current GPU implementation of the high order numerical method. Oblique inviscid shock reflection, and simulation of Rayleigh–Taylor instability.\textsuperscript{16, 17} The code is three dimensional and numerical simulations for two dimensional cases were performed by using appropriate to the scheme order number of planes in third dimension. Oblique shock reflection from a solid surface is a classical compressible code validation case while the interaction of an oblique shock with a boundary layer is a problem of current interest where LES simulations could shed light to the dynamics of the interaction. The GPUs parallelized high order method was tested for this problem first. Numerical solutions for a sequence of meshes of different sizes were obtained in order to evaluate the performance on GPUs. A numerical solution obtained with four GPUs with an 800 x 10 x 200 point mesh is shown in Fig. 3. The inflow Mach number is $M_{in}=2.9$ and the conditions at the top of the domain that generate a $M_{in}=2.9$ oblique shock at an angle 29 deg. are obtained from isentropic shock relations and correspond to a shock generated by a wedge of angle $\beta=10.94$ deg. The inflow conditions and the conditions at the top are shown in Fig. 3.

The computational domain for the simulations of the Rayleigh–Taylor instability is the box $(1 \times 0.25)$ in two dimensions and $(1 \times 0.25 \times 0.25)$ in three dimensions. The “heavy” fluid is on the left site and has density $\rho_h=2$ while the “light” fluid on the right has density $\rho_l=1$. The interface is between the two fluids is at $x=1/2$ and the variation of the initial pressure is linear throughout the domain. The initial pressure in the domain of the heavy fluid on the left is $p_h(x) = 1 + 2x$, while the variation of the pressure on the right is $p_l(x) = 1.5 + x$. The initial perturbation of axial velocity $u(y) = -0.025 \cos(8\pi y)$ is specified throughout the computational domain and the source term,\textsuperscript{16, 17} $S = (0, \rho, 0, 0, \rho u)^T$, is added for both viscous and inviscid simulations. A simulation obtained on an 400 x 10 x 100 point mesh with the 7\textsuperscript{th} order WENO scheme is shown in Fig. 4. The computed density field at time $t=4$ is shown at the lower part of Fig. 4 and the shadowgraph of the computed density field, which is proportional to the density gradient, is shown on the top. The domains interfaces are shown in Fig. 4 and it is clearly demonstrated that smiles information transfer between the domains is achieved because the variation of both field quantities and their gradients between the domains is smooth. The significant reductions of the computational time achieved through the use of GPUs, which are discussed in detail in the next section, made possible simulations of the Rayleigh–Taylor instability with different mesh densities and schemes of different order of accuracy. A comparison of two dimensional inviscid flow simulations obtained on a series of meshes is shown in Fig. 5. All simulations of Fig. 5 are for the same final time. Clearly increase of the order of accuracy yields the same effect as doubling of the mesh density in both directions. Viscous flow simulations demonstrated that grid converged solutions could be achieved for moderately high Reynolds numbers. Furthermore, use of GPUs made possible three dimensional simulations. An example of a three dimensional simulation obtained on a relatively coarse 240x60x60 point mesh is shown in Fig. 6.

B. Performance evaluation on Graphics Processors

In order to evaluate the performance of our implementation we have conducted experiments using a variety of combinations among simulation input settings and experimental platform deployment. The experimental platform consisted of a host platform supplied with a quad-core Intel Xeon X5450 processor at the clock rate of 3.0 GHz with 4GB of main memory and one NVIDIA Tesla S1070 1U computing system. The Tesla S1070 system\textsuperscript{18} makes available 4 GPUs overall, with 30 stream multiprocessors and 240 cores on each GPU, resulting on an aggregate of 960 cores for our simulations. Each GPU is equipped with 4 GB of memory.

In the following figures we present results in terms of execution times (Fig. 7) and their respective speedups (Fig. 8 and Fig. 9) that were achieved in comparison with the sequential run. The computations refer to single precision arithmetic calculations and the presented results refer to the average of 10 simulation runs of the Rayleigh–Taylor instability were each simulation conducted 100 iterations. The results for the evaluation of oblique inviscid shock reflection are similar.

According to the experimentation results the highest performance is achieved when 4 GPUs are utilized. The succeeded speedup in that case is 53 on the average for the several mesh sizes. However this specific result is lower
than the optimal if we compare it with the average speedup that we achieve when we utilize a single Tesla GPU. This is mainly due to the data transfers that have to take place between the host memory and the several device memories on each time step of the simulation. Nevertheless the speedup is still significant compared to the sequential or the OpenMP executions.

V. Conclusions

A high order WENO finite difference method was implemented for parallel processing with GPUs. A two level domain decomposition approach was employed in order to achieve highly accelerated processing. On the first level the domain was decomposed into subdomains that are assigned to multiple GPUs. On the second level within each GPU the domain was further decomposed into an appropriate number of thread blocks. High resolution simulations of oblique shock reflection and Rayleigh–Taylor instability were used as computational examples for the evaluation of the parallelization efficiency. It was found that a significant, however yet suboptimal, speedup was achieved with the available number of GPUs. It is anticipated that better handling of data transfer among GPUs can further increase processing speed to optimal levels. It is also expected that implementation of the existing parallelization approach on clusters with GPUs would make possible large scale DNS and LES computations.

References

Figure 3. Detail of shock reflection at $M=2.9$ computed on 4 GPUs

Figure 4. Simulation of Rayleigh–Taylor instability computed on 4 GPUs using a 400 x 100 point mesh.
Figure 5. Effect of grid (h – type) refinement and order of accuracy (p – refinement) on the resolution of the Rayleigh–Taylor instability

Figure 6. Numerical simulation of Rayleigh–Taylor instability using a 240 x 60 x 60 point mesh.
Figure 7. Execution times for several grid sizes

Figure 8. Speedup of OpenMP and GPU implementations compared to sequential execution I

Figure 9. Speedup of OpenMP and GPU implementations compared to sequential execution II