CUDA UNIFIED MEMORY ACCELERATION OF THE PROCESS OF DEVELOPING MULTI-GPU APPLICATIONS

A Y Gorchakov 1,2,3 and A V Solovjev 2

1 Dorodnicyn Computing Centre, Federal Research Center “Computer Science and Control” of Russian Academy of Sciences, Moscow, Russia
2 Nuclear Safety Institute of the Russian Academy of Sciences, Moscow, Russia
3 Moscow Institute of Physics and Technology, Dolgoprudnyi, Moscow oblast, Russia

andrgor12@gmail.com

Abstract. The experience of using Cuda Unified Memory technology to develop Multi-GPU applications is described. The paper deals with the method of transferring computations on the MGDS-GPU model two-dimensional problem described by the equations of single-layer shallow water with variable density over an uneven bottom. To solve the problem was applied balance - characteristic scheme "CABARET". The minimum steps necessary to go from computing on the GPU to computing on the MULTI-GPU are given. The computational efficiency is investigated depending on the size of the grid.

Introduction

The solution of modern practical problems of mathematical modeling, problems of assimilation of measurement data requires the use of high-performance computing. In this case, the typical performance of the computing complex should be tens to hundreds of TFlops or more. Such characteristics are achieved either on clusters equipped with thousands of CPUs, or on single servers with several graphic accelerators like Nvidia Tesla V100 (for reference, the performance of this graphic accelerator is 7.8 teraflops during double precision operations). The Message Passing Interface (MPI) standard is used in software development. Using MPI, the researcher instead of solving purely mathematical problems begins to solve technical problems, for example, the problem of choosing the type of messaging – blocking/non-blocking (MPI_send/MPI_isend), or the problem of choosing algorithms for collective operations. If we move on to computing on graphic accelerators, then for calculations on a single graphics accelerator, we forget about these problems, but when moving to Multi-GPU calculations, again the question arises how to exchange data between devices.

In this work, we investigated the possibility of using Cuda Unified Memory technology for calculations on several graphics accelerators. The main idea of the technology is the use of a single address space, which includes both the entire system memory and the memory of all installed graphics accelerators (supporting CUDA). To allocate unified-memory, it is enough to call the cudaMallocManaged function, and then without any effort from the developer, the system using the Page Migration engine (starting with the Pascal architecture and further), independently organizes the
exchange of CPU-GPU/GPU-GPU data. A side effect of using unified-memory is to increase the available memory on the GPU.

1. Statement of the model problem
To test the effectiveness of Cuda Unified Memory, a relatively simple model two-dimensional problem is considered [1]. The problem contains the main features of this class of problems, namely, the problem of fluid dynamics with a free surface and a variable density in the approximation of a single-layer shallow water over an uneven bottom. Usually shallow water equations are written for the case of constant density, and in this case we consciously go to the complexity of the problem to bring it closer to reality.

The system of differential equations describing the dynamics of single-layer shallow water with variable density follows from the laws of conservation of mass, momentum, and continuity equations and has the form:

\[
\frac{\partial p}{\partial t} + \frac{\partial (p u)}{\partial x} + \frac{\partial (p v)}{\partial y} = 0; \quad \frac{\partial (p u)}{\partial t} + \frac{\partial (p u^2)}{\partial x} + \frac{\partial (p u v)}{\partial y} + g \frac{\partial (h^2)}{\partial x} = -g \rho h \frac{\partial B}{\partial x};
\]

\[
\frac{\partial (p v)}{\partial t} + \frac{\partial (p u v)}{\partial x} + \frac{\partial (p v^2)}{\partial y} + 2 \frac{\partial (p u v)}{\partial y} = -g \rho h \frac{\partial B}{\partial y}; \quad \frac{\partial h}{\partial t} + \frac{\partial h u}{\partial x} + \frac{\partial h v}{\partial y} = 0;
\]

(1.1)

Here \( \rho \) - the density, \( u,v \) - the velocity components along the directions \( x,y \), respectively, \( B(x,y) \) - the bottom relief function, \( h = H - B \) - the thickness of the liquid layer, \( H(x,y,t) \) - the level of the free surface. From the first and last equations of this system follows the density transport equation.

Let the area \( G \) in which the problem is considered has the form of a rectangle with sides \( L_x, L_y \) (Figure 1).

![Figure 1 – Area G](image)

![Figure 2 - scheme "CABARET"](image)

The boundaries of this region will be denoted by \( S_R, S_L, S_T, S_B \), which corresponds to the right, left, upper and lower sides of the rectangle. The bottom relief function is chosen as a piecewise constant:

\[
B(x,y) = \{ \Delta z, \; \text{if} \; (x,y) \in G^* \; \text{and} \; 0, \; \text{if} \; (x,y) \in G / G^* \}.
\]

(1.2)
As the boundary conditions on the sides \( S_R, S_L \), we set: \( u\big|_{S_R} = u\big|_{S_L} = 0 \).

On the part of the upper boundary \( S_T^* \), we define the "upper reservoir" condition, on the lower boundary part \( S_B^* \) - the "lower reservoir" condition, which determine the values of the Riemann invariants coming from outside the region:

\[
H^*_T = \text{const} > H(x, y, t_0); u_T^* = v_T^* = 0; \rho_T^* = f(t); (x, y) \in S_T^*
\]

\[
H^*_B = \text{const} < H(x, y, t_0); u_B^* = v_B^* = 0; \rho_B^* = \rho_0; (x, y) \in S_B^*
\]

From the "upper reservoir" liquid flows into the region \( G \), and from the region \( G \) it flows into the "lower reservoir". On the remaining sections of the upper and lower boundaries, the non-percolation condition is set:

\[
v\big|_{S_T^*, S_B^*} = v\big|_{S_T^*, S_B^*} = 0
\]

We set the initial data as follows. We assume that at the initial moment \( t_0 \) the liquid is at rest, has a constant density, and its free surface is horizontal, and

\[
H(x, y, t_0) = H_0; \quad H_T^* > H_B > \Delta z; \quad H_0 > H_B^*; \quad \Delta z < H_B^*
\]

The direct problem is solved until the time \( t_1 \) at which the quasi-steady vortex flow can be considered to be formed. On this time interval, the density of the inflowing liquid is considered equal to \( \rho_0 \). Then, another density \( \rho_1 > \rho_0 \) is applied to the input and a dynamic substitution of one liquid for another takes place. We assume that the values of the initial parameters are given in such a way that the flow remains subsonic. In the second time interval, a discrete spatial-temporal monitoring network \( \Theta_m \) is specified that fixes the density, velocity components and the height of the free surface \( \{\rho_k, u_k, v_k, H_k\}, k \in \Theta_m \).

The task of four-dimensional assimilation of monitoring data consists in determining the values of the density, velocity, and height of the free surface from these values at the instant of time \( t_1 \).

2. Algorithm for the numerical solution of a direct problem

For the numerical solution of a direct problem, we use the CABARET scheme [2], combining the merits of conservative and characteristic methods. The area \( G \) is covered by a regular grid with rectangular cells with sides \( \Delta x, \Delta y \) (Figure 2). The calculated physical fields will be referred to both the centers of the calculated cells \( \{\rho_C, u_C, v_C, H_C\} \) and to the centers of their faces \( \{\rho, u, v, H\} \), where the asterisk takes on values \( R, L, T, B \). The values relating to the centers of the cells will be called "conservative", and to the midpoints of the faces – "fluxshape".

In the CABARET scheme three consecutive stages are distinguished - three phases. In the first phase, conservative variables are calculated on the intermediate time layer:

\[
\frac{1}{2} \left[ \psi_{c}^{n+1/2} - \psi_{c}^{n} \right] + \bar{A} \left( \varphi_{f}^{n} \right) = 0
\]

where \( \psi_{c}^{n} = \left[ (h), (\rho h), (\rho hu), (\rho hv) \right]^{T} \), \( \bar{A} \) is the operator approximating the spatial derivatives in (1.1), the small index \( f \) indicates fluxshape variables on the faces of the cell. Conservative variables \( \{h_C^{n+1/2}, \rho_C^{n+1/2}, u_C^{n+1/2}, v_C^{n+1/2}\} \) are evaluated sequentially explicitly.

Phase 2 calculates the values of fluxshape variables on a new time layer \( t_{n+1} \). At this stage, the characteristic forms of equations (1.1) along the axis \( x \) are used:
A similar equation can be written along the axis \( y \). Here \( c = \sqrt{gh} \) is the speed of the sound, the right-hand parts \( D^*_k, D_{L}^*_k; k = 1, 2, 3, 4 \) are fairly complex expressions that depend on the derivatives along the orthogonal direction. The form of these expressions in the CABARET scheme is not used.

It should be noted that the characteristic equations (2.2) in contrast to shallow water equations at constant density, can not be represented in total differentials and there are no Riemann invariants for the entire flow. However, when going from a differential problem to a discrete one for each computed space-time cell, you can enter a so-called "Local" Riemann invariants:

\[
I_1^* = u + h \cdot \left( \frac{g}{c} \right)_{c}^{n+1/2} + \rho \cdot \left( \frac{c}{2\rho} \right)_{c}^{n+1/2} ; \quad I_2^* = u - h \cdot \left( \frac{g}{c} \right)_{c}^{n+1/2} - \rho \cdot \left( \frac{c}{2\rho} \right)_{c}^{n+1/2} ; \quad I_3^* = u; \quad I_4^* = \rho ;
\]

In terms of these expressions, the characteristic equations (1.8) are approximated as follows:

\[
\frac{(I_1^*)_{c}^{n+1/2} - (I_1^*)_{c}^{n}}{\tau/2} + \frac{(\lambda_{k})_{c}^{n+1/2} \cdot (I_3^*)_{c}^{n} + (I_3^*)_{c}^{n}}{\Delta x} = \{D^*_k\}_{c}^{n+1/2} + O(\tau, \Delta x^2) ; \quad k = 1, ..., 4 ;
\]

Similar formulas are obtained in the direction \( y \). After the computation of phase 1, the conservative variables on the intermediate time layer are already known to us, therefore the left-hand sides of formulas (2.4) are also known and serve as a definition for the unknown right-hand sides.

Calculation of the values of local Riemann invariants on the new time layer is carried out by linear extrapolation. For the first two invariants in subsonic flows we obtain:

\[
\begin{align*}
(I_1^*)_{c}^{n+1} &= 2(I_1^*)_{c}^{n+1/2} - (I_1^*)_{c}^{n} ; \quad (I_2^*)_{c}^{n+1} = 2(I_2^*)_{c}^{n+1/2} - (I_2^*)_{c}^{n} ; \\
(I_1^*)_{r}^{n+1} &= 2(I_1^*)_{r}^{n+1/2} - (I_1^*)_{r}^{n} ; \quad (I_2^*)_{r}^{n+1} = 2(I_2^*)_{r}^{n+1/2} - (I_2^*)_{r}^{n}.
\end{align*}
\]

The direction of extrapolation for the third and fourth invariants depends on the direction of their transference speed on the intermediate time layer, which on each edge is assumed to be equal to half the sum of the conservative velocities in the adjacent cells:

\[
\begin{align*}
(I_3^*)_{c}^{n+1} &= (I_3^*)_{c}^{n+1/2} - (I_3^*)_{c}^{n} ; \quad (I_4^*)_{c}^{n+1} = (I_4^*)_{c}^{n+1/2} - (I_4^*)_{c}^{n} \quad \text{if \( u_{c}^{n+1/2} > 0 \)} ; \\
(I_3^*)_{r}^{n+1} &= (I_3^*)_{r}^{n+1/2} - (I_3^*)_{r}^{n} ; \quad (I_4^*)_{r}^{n+1} = (I_4^*)_{r}^{n+1/2} - (I_4^*)_{r}^{n} \quad \text{if \( u_{r}^{n+1/2} \leq 0 \)} ;
\end{align*}
\]

Thus, four local invariants are calculated for the vertical faces of the calculation cells located within the computational domain. Similar formulas are obtained for horizontal faces \( T, B \).

For the faces lying on the boundary of the domain, at best, three invariants will be computed. The missing invariant is replaced by the boundary condition - the vanishing of the normal component of the velocity vector on an impermeable wall, or by a local inverting coming outside the region from the "upper" or "lower" reservoir.

A very important operation in the CABARET scheme, as in all other high resolution schemes, is the non-linear correction of flows, which prevents the appearance of nonphysical oscillations in the solution. Here it is implemented on the basis of the maximum principle, which must obey local invariants - their values should be in the interval between the maximum and minimum, determined by the expressions:

\[
\begin{align*}
\max (I_k^*) &= \max \{I_{L}^*, I_{C}^*, I_{R}^*\} + \tau \cdot (D^*_k)_{c}^{n+1/2} ; \quad \min (I_k^*) = \min \{I_{L}^*, I_{C}^*, I_{R}^*\} + \tau \cdot (D^*_k)_{c}^{n+1/2} ; \quad k = 1, ..., 4 ;
\end{align*}
\]

Nonlinear correction consists in the fact that if the value of the local invariant goes beyond the interval defined in this way, then the value of the nearest boundary of this interval is assigned to it.
After this, according to the corrected invariants, the values of the fluxshape variables on the new time layer are calculated.

The process of calculating one time step is completed by **phase 3**, where new values of conservative variables are found.

\[
\frac{\tilde{\psi}^{n+1}_c - \tilde{\psi}^{n+1/2}_c}{\tau/2} + \tilde{\lambda} \left( \tilde{\varphi}^{n+1} \right) = 0; \quad \tilde{\psi}^{n+1}_c = \left[ \left( h, (\rho h), (\rho hu), (\rho hv) \right) \right]^{n+1}_c
\]

The CABARET difference scheme belongs to the class of high resolution schemes and has the second order of approximation on smooth solutions, when the flow correction procedure does not "work". It allows to resolve of thin vortex flow patterns on relatively coarse computation grids and is well parallelized.

3. **The numerical experiment**

To perform the calculations, the computing resources of FIC IU RAS were used [3] - 2xIBM Power9 3.8 Hz, 1024 GB RAM, 4xNvidia Tesla V100, 16GB RAM. The Nvidia CUDA Toolkit 9.2 was used. The area was covered with a regular grid $N \times N$ - $N$ cells along the x axis, $N$ cells along the y axis. Where $T$ took the values 500, 700, 1000, 1400, 2000, 2800, 4000. The height of the liquid in the "upper" reservoir $H^*_{t} = 1.1$, the height of the liquid in the "lower" reservoir $H^*_{b} = 0.9$, the density of the inflowing liquid $\rho_0 = 1$ and $\rho_1 = 1.1$. Parameters of the bottom relief function $B(x, y) - \Delta z = 0.3$, $G^*$ - a rectangle located in the center of the area $G$ with sides $0.4L_x, 0.4L_y$ (see Figure 1).

Memory allocation was carried out by the function cudaMallocManaged. It should be noted that by default, the data exchange between graphics accelerators goes along the path GPU-CPU-GPU, in order to allow the exchange of GPU-GPU, we used the function cudaDeviceEnablePeerAccess. For each graphics accelerator, a stream was created through the cudaStreamCreate function. The regular grid is divided into 4 parts along the x-axis, according to the number of graphic accelerators. The kernels corresponding to each phase of the CABARET scheme were launched for each of the 4 areas of the regular grid in the corresponding stream. Kernel synchronization was carried out through the event mechanism (cudaEventSynchronize). The results of acceleration compared to calculations on one graphics accelerator are presented in Figure 3.

![Figure 3 - acceleration of calculations depending on the size of a regular grid](image-url)
As can be seen from figure 3 for small grids, we do not get acceleration. This is due to the fact that the exchange is not implemented effectively enough, and is made page by page. The page size, according to information from the Unified profiling memory, is 64 KB result. Increasing the size of the grid we come to an acceleration of four times.

**Conclusion**

The use of technology Cuda Unified Memory allows without the application of any significant labor costs to move to the calculations on Multi-GPU systems. In some cases, you can get a performance boost. At the moment, this technology is advisable to use for calculations on single servers with large grids. It can be expected that the further development of this technology will expand the scope of its application.

**Acknowledgments** The authors express their gratitude to the postgraduates Mayorov Pavel and Mayorov Petr for their assistance in the calculations.

The work was supported by the RSCF, project No. 18-11-00163.

**References**

[1] Evtushenko Y. G., Gorchakov A. Y., Goloviznin V. M. Fast automatic differentiation in problems variations four-dimensional data assimilation (4Dvar) //Journal of Physics: Conference Series. – IOP Publishing, 2018. – T. 1128. – №. 1. – C. 012001.

[2] Goloviznin V M, Zaytsev M A, Karabasov S A and Korotkin I A 2013 New algorithms of computational hydrodynamics for multiprocessor computer systems (Moscow: Moscow University)

[3] Federal Research Center “Computer Science and Control” of Russian Academy of Sciences [Electronic resource]: site. – (Moscow: FRC CSC RAS) - URL: http://frccsc.ru